VŠB – Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra kybernetiky a biomedicínského inženýrství

Návrh nástroje pro data mining v biomedicínské oblasti

Proposal for a Tool for Data Mining in the Biomedical Field

2015 Tereza Vicherek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání diplomové práce

Student: **Bc. Tereza Vicherek**

Studijní program: N2649 Elektrotechnika

Studijní obor: 3901T009 Biomedicínské inženýrství

Téma: Návrh nástroje pro data mining v biomedicínské oblasti
Proposal for a Tool for Data Mining in the Biomedical Field

Zásady pro vypracování:

1. Současné nástroje pro data mining nad databázemi typu MySQL a SQL.
2. Rozbor problematiky návrhu grafického prostředí pro data mining nad medicínskými daty.
3. Návrh grafického variabilního prostředí pro data mining nad medicínskými daty.
4. Implementace navrženého prostředí.
5. Odzkoušení nástroje na již vytvořených databázích.
6. Zhodnocení dosažených výsledků.

Seznam doporučené odborné literatury:

[1] DUBOIS, Paul. *Mysql Cookbook Solutions for Database Developers and Administrators.* 3. Aufl. Sebastopol: O´Reilly, 2014. ISBN 978-1-449-37402-0.
[2] WITTEN, I., E. FRANK a M. A. HALL. *Data mining: practical machine learning tools and techniques.* 3rd ed. Amsterdam: Morgan Kaufmann, 2011, xxxiii, 629 s. Morgan Kaufman series in data management systems. ISBN 978-0-12-374856-0.
[3] EPSTEIN, Irwin a Susan BLUMENFIELD. *Clinical data mining in practice-based research: social work in hospital settings.* New York: Haworth Social Work Practice Press, c2001, xxiv, 190 p. ISBN 0-7890-1709-1.
[4] GROSSMAN, R.L. et al.(Eds.)*Data mining for scientific and engineering applications.* Dordrecht: Kluwer Academic, 2001. ISBN 978-1-4615-1733-7 (e-Book)ISBN 978-1402001147.
[5] McCULLOUGH, Jeffrey. *Transfusion medicine.* 3rd ed. Chichester, West Sussex, UK: Wiley-Blackwell, 2011. ISBN 1-4443-3705-X.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. David Vala**

Datum zadání: 01.09.2014
Datum odevzdání: 07.05.2015

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * OSTRAVA *

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracovala samostatně. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

31. 7. 2015

Datum

Podpis

Poděkování

Ráda bych poděkovala Ing. Davidu Valovi a Ing. Janě Nowakové za jejich vedení a pomoc při tvorbě této diplomové práce. Dále pak zaměstnancům Krevního centra FN Ostrava, zvlaštepak paní Ing. Dagmaře Valové, Ing. Jaroslavu Mourečkovi a Mgr. Petře Kovářové, za jejich čas a ochotu. Chtěla bych také poděkovat manželovi, za jeho technickou a odbornou pomoc.

„Souhlasím se zveřejněním této bakalářské/diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava."

V Ostravě 28. 7. 2015

Abstrakt

Každým měsícem, ne-li každým týdnem, nalézáme uplatnění data miningu v nových oblastech života, od podnikání po výzkum, od akademie po státní správu. Tato práce analyzuje koncepty data miningu z pohledu biomedicínských dat a přináší nástroj naplňující konkrétní potřeby v biomedicínské oblasti.

Klíčová slova

Data mining, databáze, SŘBD, MySQL, SQL, GUI nástroje

Abstract

Every month, if not every week, data mining finds its way through to a new area of business and research, study and government. This work analyzes concepts of data mining from the perspective of biomedical data, and delivers a tool adapted to a specific need in the biomedical community.

Key words

Data mining, Database, RDBMS, MySQL, SQL, GUI tools

Obsah

1 Úvod

Data mining je významnou oblastí aktivit disciplíny business intelligence, který je schopen významnou měrou přispět k efektivním rozhodnutím v řízení účelových aktivit velkého rozsahu (podniků), ať už jsou orientovány komerčně (firma), vědecky (vědecký ústav) nebo společensky (nemocnice).

Data mining se zabývá nalézáním konkrétních a specifických odpovědí na otázky, jejichž podstata většinou tkví ve velkém a často nepřehledném množství dat a bez náležitého uchopení takových dat nelze příslušné odpovědi nalézt.

Proto i nástroje pro data mining hrají v takových rozhodnutích klíčovou roli, i když jsou jen jedním z několika článků nezbytných pro úspěšné uplatňování shromážděných dat.

V této diplomové práci nejprve rozebírám problematiku databází, které jsou v jádru každé data miningové aktivity.

V dnešní společnosti informačního věku, kdy, jednoduše řečeno, je všechno se vším informačně propojeno, je často nezbytné také ochraňovat soukromí osob, kterých se data mohou týkat. Taková ochrana byla nezbytná i pro praktickou část této diplomové práce. Následně se tedy zabývám touto ochranou v kapitole o anonymizaci.

Přehledem některých současných data mining nástrojů pak otevírám titulní téma této diplomové práce.

Každý moderní interaktivní počítačový nástroj, který má pomáhat lidem vytvářet trvalé hodnoty, se neobejde bez kvalitního grafického prostředí. To samozřejmě platí i pro data mining nástroje. Co takové prostředí vyžaduje popisuji v následné kapitole.

Tímto pak přecházím v praktickou část diplomové práce.

Naskytla se mi možnost uplatnit mnohé nashromážděné poznatky v praxi, a to dodáním Krevnímu centru při fakultní nemocnici v Ostravě grafického nástroje pro intuitivní data mining, tedy těžbu dat nashromážděných v jejich mnohaleté operační databázi, obsahující přes 60 miliónů záznamů různých typů. Tuto možnost jsem využila k širokému uplatnění databázových vědomostí i teoretických znalostí v oblasti anonymizace.

Výsledkem je nástroj, který tak završil propojení výše uvedených poznatků s vědeckými a do nějaké míry i každodenními aktivitami Krevního centra.

Ač mne tato praktická část zavedla k mnohem detailnějšímu poznání některých technologií, než jsem původně pro rámec této diplomové práce zamýšlela, výsledný efekt byl podstatným přínosem jak pro Krevní centrum, tak i pro mne.

TEORETICKÁ ČÁST

2. Databáze

Databáze je uspořádaná množina informací (dat). V širším smyslu můžeme slovo databáze použít i pro software, který umožňuje manipulaci s uloženými daty a přístup k nim. Tento software se v české literatuře nazývá nejčastěji systém řízení báze dat (SŘBD). Běžně se však pod označením databáze myslí jak uložená data, tak i software pro přístup k těmto datům.

2.1 Důležité pojmy v souvislosti s databázemi

Systém
Systém můžeme chápat jako množinu prvků a vazeb mezi nimi.
Systém je agregací podobných nebo alespoň vzájemně souvisejících jevů, věcí, procesů a souboru pravidel pro jejich jednání (fungování). (Bartalanffy)

Data
Data jsou tvarem množného čísla latinského slova datum, které můžeme přeložit jako “něco daného” a které bylo původně odvozeno ze slova dare, neboli dát. V kontextu počítačové vědy se pod pojmem data vždy používal jako označení pro čísla, text, zvuk, obraz, popř. jiné smyslové vjemy reprezentované v podobě vhodné pro zpracování počítačem. [1] Data jsou údaje získaná pozorováním, měřením apod.

Informace
Informace jsou pouze taková data, která nám mohou být k něčemu užitečná. Taková, která se dají rozumně interpretovat.

Systém řízení báze dat
Systém řízení báze dat (SŘBD) je sofware, které zajišťuje práci s databází. Tvoří takzvané rozhraní mezi aplikačními programy a uloženými daty. Jeho úlohou je efektivně pracovat s velkým množstvím dat, musí být schopen data ukládat, modifikovat, mazat a provádět dotazy. U operačních databází je kladen důraz na efektivitu schopnosti data ukládat, modifikovat, mazat a u warehousových pak na efektivitu schopnosti provádět dotazy.

Báze dat
Báze dat je množina vzájemně propojených dat, které využívají aplikace.

Databázový systém
Databázový systém můžeme popsat následující rovnicí:
Databázový systém = Systém Řízení Báze Dat + Báze dat

Datové entity
Objekty v databázi – tabulky, indexy, procesy, pohledy apod.

Atribut
Atribut neboli položka je jednotlivý sloupec v tabulce.

Záznam
Záznamem se označuje jednotlivý řádek v tabulce.

Primární klíč
Primární klíč je atribut, jehož hodnota je pro každý záznam jedinečná.

Cizí klíč
Slouží pro vyjádření vztahů (relací) mezi databázovými tabulkami. Umožňuje identifikovat, které záznamy z různých tabulek spolu navzájem souvisí.

2.2 Historie databází

Jako prvního předchůdce databází bychom mohli označit papírovou kartotéku. Kartotéky umožňovaly uspořádání dat podle různých kritérií a vkládání nových položek. Veškeré operace prováděl přímo člověk. Správa těchto kartoték by se v mnohém dala přirovnat ke správě dnešních databazí.
V roce 1890 vytvořil Herman Hollerith první automat na bázi děrných štítků. Herman Hollerith se v roce 1911 spojil s další firmou a vznikla nová firma International Business Machines (IBM).
Na počátku databázové éry byly velkým zájemcem o databázové systémy státní úřady USA. Během první světové války používaly úřady systém děrných štítků. Když byla v roce 1935 v USA uzákoněna nutnost vedení informací o přibližně 26 milionech zaměstnancích, vytvořila IBM nové zařízení. Vznikl tak první digitální počítač pro komerční použití – tzv. UNIVAC I., který byl založený na státem podporovaném projektu Electronic Discrete Variable Automatic Computer (EDVAC) z University of Pennsylvania. V roce 1959 disponoval Pentagon již více než 200 počítači. Samozřejmě se stále jednalo o počítače pracující na principu děrných štítků.
Ve stejném roce se konala konference zástupců firem, uživatelů a amerického ministerstva obrany, jejímž závěrem byl požadavek na univerzální databázový jazyk.
V roce 1960 vzniklo uskupení Data Systems Languages (Codasyl), které bylo ustanoveno ministerstvem obrany USA, aby standardizovalo softwarové aplikace. Výsledkem pak byl common business-oriented language (COBOL). Dalším pokrokem byl přechod od magnetických pásek, které

umožňovaly jen sériový přístup k datům, k magnetickým diskům.
Charles Bachman z General Electric v roce 1961 představil první integrovaný datový sklad, který obsahoval první náznak databázového managemantu. O několik let později pak Bachman a další výzkumníci založili v rámci uskupení Codasyl samostatnou skupinu, která se měla věnovat vývoji databázových systémů (Database Task Group – DBTG). Tato skupina poté publikovala základní specifikaci pro programovací jazyky určené pro práci s databázemi – zvláště pak COBOL. Na bázi specifikace od Codasylu následně vznikla řada produktů od firem Eckert-Mauchly Computer Corporation, Honeywell Incorporated, Siemens AG a pro mikropočítače od firem Digital Equipment Corporation (DEC) a Prime Computer Corporation.
Codasyl produkt měla také IBM, která ho uvedla na trh v roce 1968 pod názvem IMS. Tento systém vznikl jako odvozenina od projektů vzniklých během projektu Apolo v NASA a pracoval na počítači System/360. Většina Codasyl kompatibilních databází používala síťový model dat, zatímco IBM použila u své implementace model hierarchický. Jeden ze zaměstnanců IBM, který do této firmy IBM vstoupil v roce 1949 a jehož jméno bylo Ted Codd, byl nespokojený jak s Codasylem, tak s vlastní implementací od IBM. Tento muž v roce 1970 publikoval článek „A Relational Model of Data for Large Shared Data Banks“, ve kterém navrhoval implementaci nového datového modelu, který byl nazván relačním.
Ted Codd také navrhl možnost, jak použít relační kalkul a algebru i pro netechnické uživatele při ukládání a manipulaci s daty. Hlavní výhodou tohoto nového modelu je používání srozumitelných příkazů vycházejících z běžné angličtiny. Už tato původní koncepce předpokládala ukládání dat do tabulek. Tento návrh byl založen na nezávislosti dat na použitém hardware, na způsobu fyzického uložení a také na přístupu k datům pomocí ne-procedurálního jazyka.
Dle relační teorie lze pomocí pěti základních operací uskutečnit veškeré operace s daty. Tyto operace jsou sjednocení, kartézský součin, rozdíl, selekce, projekce a spojení. Ostatní operace jsou jen kombinace těchto pěti.
Tento návrh byl z počátku vnímán jako kuriozita a ani IBM se nechtěla vrhnout do implementace něčeho, co by ve své podstatě zavrhlo její produkt IMS. Ted Codd z tohoto důvodu publikoval svůj nový projekt také mimo IBM.
Po různých debatách nakonec vznikly dva projekty relačních databází, a to System-R pod křídly IBM a od roku 1972 projekt Ingres na University of California at Berkeley (UC-Berkeley) s podporou armády a National Science Foundation (NSF).
Vývoj relačních databází u IBM prošel dvěmi fázemi. V první fázi v období 1974-1975 šlo hlavně o ověření návrhu relačního modelu. V druhé fázi v letech 1978-1979 se již implementoval plně funkčí více uživatelský systém. Původní jazyk System-R byl SEQUEL (Structured English Query Language). V listopadu roku 1976 pak byla v IBM popsána verze SEQUEL2, která byla poté přejmenována na SQL. Časem byl tento jazyk uznán jako standard. I přesto se však do této doby nepodařilo management IBM přesvědčit o možnosti nahrazení staré IMS technologie produktem na bázi relací.
Druhá větev vývoje byla vedena na University of California at Berkeley dvojicí Michael Stonebraker a Eugene Wong. Jejich cílem bylo vytvoření systému pro geografická data Berkeleyské ekonomické

fakulty. Zde se pracovalo na počítačích PDP-11/45. Výsledkem jejich práce byl Ingres (Interactive Graphics and Retrieval System).
Ingres i System-R byly vyvíjeny pod různými operačními systémy a různým hardware. Ingres nepoužíval SQL, ale velmi podobný jazyk QUEL. Postupně byl Ingres uvolněn asi v tisícovce kopií a Michael Stonebraker založil Ingres Corporation pro komercializaci kódu Ingresu. Další ze členů Berkeley týmu Robert Epstein odešel a stal se spoluzakladatelem Sybase. Komerční verze Ingresu byla dostupná v roce 1980. Další ze členů Ingres týmu Paula Hawthorn odešel a podílel se na založení Illustra Information Technologies Incorporated, v současné době známé jako Informix.
Lze říci, že kód z Berkeley se stal inspirací nebo přímo zdrojem k založení téměř všech dnes známých SQL databází. Tým pracující na původním Ingres neměl nikdy více než pět až šest programátorů, u IMB pracovalo na System-R 15 programátorů. Mezi oběma týmy existovala spolupráce a Berkeleyská univerzita posílala studenty na stáže do IBM.
V roce 1980 byl první SQL databází na trhu Oracle pro počítače VAX. Firmu Oracle založil Larry Ellison, který se nechal inspirovat v System-R od IBM. Po určitých pochybnostech přišla na trh i IBM se svým produktem DB2 (pro počítače MVS).
Výzkum v oblastech relačních databází byl prováděn i na jiných místech než jen u IBM a Berkeley. Určitý podíl mají University of Toronto, University of Utah, University of Wisconsin.
V roce 1987 ISO a ANSI publikovali standard SQL a v roce 1989 publikuje ISO dodatek Integrity Enhancement Feature, začíná se hovořit o SQL89. Tento ISO standard je v roce 1992 upraven a dále se o něm hovoří jako o SQL2 nebo SQL92.
V roce 1982 byl ukončen původní projekt Ingres a v roce 1985 byl předělán do projektu Postgres. Úkolem Postgresu bylo vytvořit relačně-objektový databázový systém. Projekt Postgres trval do roku 1994 a v jeho čele byl opět Michael Stonebraker. Po ukončení projektu se v letech 1994-1995 dva studenti univerzity Berkley - Jolly Chen a Andrew Yu – rozhodli pracovat na akademickém kódu Postgresu, který přejmenovali na Postgres95. V létě roku 1996 přechází Postgres95 do podoby open source a je přejmenován na PostgreSQL, který je ve vývoji dodnes.
Kalifornská univerzita v Berkeley dále částečně pokračovala ve vývoji původního Postgresu, a to v armádou podporovaném projektu nazvaném Mariposa.
Jeden z největších databázových systémů byl vybudován v CERNu. [2]

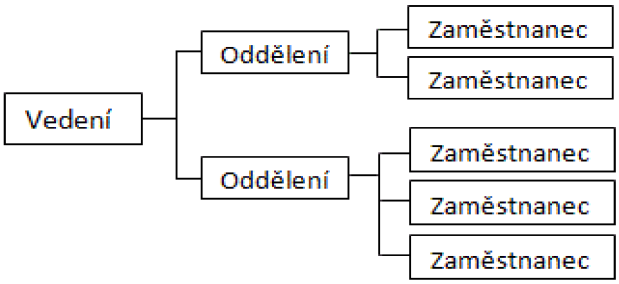
2.3 Databázové modely

Databázový model je kolekce pojmů, na kterých je vybudován jazyk pro definici dat. Databázový model není tedy výsledek modelování, ale prostředek modelování. Výsledkem jeho aplikace je schéma databáze. [3]

2.3.1 Hierarchické databáze

Data jsou strukturována hierarchicky a obvykle jsou znázorňována v podobě větví/kořene stromů. Je to první datový model, který byl v minulosti hojně využíván. Nejznámějším hierarchickým

systémem řízení báze dat byl IMS od společnosti IBM. Největší nevýhodou hierarchického uspořádání je složitá operace vkládání a rušení záznamů a v některých případech i nepřirozená organizace dat. Vzhledem k těmto nedostatkům bylo časem od jeho používání upuštěno a zaměněno za koncepci síťových a později ralačních databází.

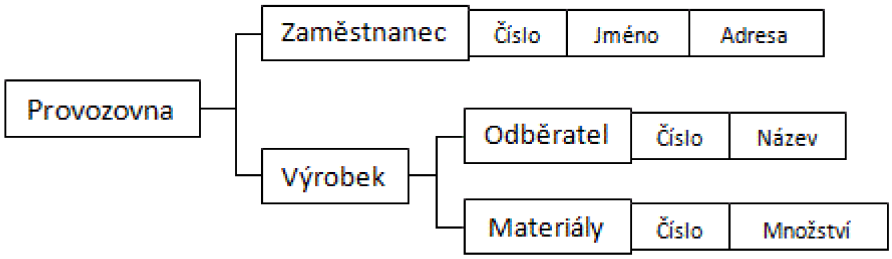


Obr.1 Hierarchická struktura databází [4]

Vztah v databázi je reprezentován termíny rodič a potomek. V tomto typu vztahu může být tabulka rodiče přidružena k jedné, nebo více tabulkám potomků, ale tabulka potomka může být přidružena pouze k jedné tabulce rodiče.

2.3.2. Síťové databáze

Síťová databáze byla vyvynuta hlavně jako pokus o vyřešení problémů hierarchické databáze. Síťový model je v podstatě zobecněním hierarchického modelu, který je doplněn o mnohonásobné vztahy (sety). Tyto sety propojují záznamy různého či stejného typu, přičemž spojení může být realizováno na jeden nebo více záznamů. Přístup k propojeným záznamům je přímý bez dalšího vyhledávání. Nevýhodou síťové databáze je zejména nepružnost a obtížná změna její struktury. [4] Síťové databáze převládaly v komerčních databázových systémech v 80. letech 20. století. Nejznámějším prodektem byl IDMS od firmy Culliname Corporation.



Obr. 2 Síťový model dat [4]

2.3.3 Relační databáze

Základem každé ralační databáze jsou relace (databázové tabulky), což jsou dvourozměrné struktury tvořené záhlavím a tělem, které slouží k přímému uložení dat do paměťového prostoru relačné databáze. Jejich sloupce se nazývají atributy. Atributy mají určen svůj konkrétní datový typ a doménu, což je množina přípustných hodnot daného atributu. Každý sloupec musí mít také jedinečný název. Další částí tabulky jsou řádky nebo také záznamy. Oba tyto pojmy jsou identické, jelikož jeden řádek reprezentuje jeden záznam. Pro většinu účelů by každý řádek měl mít určitý jedinečný identifikátor, který jednoznačně určí příslušný záznam. Tento problém řeší klíče.
Kandidátní klíč je atribut nebo skupina atributů, které jednoznačně identifikují záznam v relační tabulce. Kandidátní klíč se může stát primárním klíčem. Ty, které se primárním klíčem nestanou, jsou označovány jako alternativní klíče.
Primární klíč je jednoznačný identifikátor záznamu, řádku tabulky. Primárním klíčem může být jediný sloupec nebo kombinace více sloupců tak, aby byla zaručena jednoznačnost. Pole klíče musí obsahovat hodnotu, to znamená, že se zde nesmí vyskytovat nedefinovaná prázdná hodnota.
Výhoda ralačních databází je zejména v tom, že jsou relativně jednoduché a snadno pochopitelné. Což je také důvod, proč jsou rozšířené a mezi programátory oblíbené. [5]

Vztahy mezi tabulkami:

Relace slouží ke svázání dat, která spolu souvisejí a jsou umístěny v různých databázových tabulkách. V zásadě rozlišujeme čtyři typy vztahů:

- mezi daty není žádná spojitost, proto nedefinujeme žádný vztah
- 1:1 – záznamu odpovídá právě jeden záznam v jiné databázové tabulce a naopak
- 1:N – jednomu záznamu přiřazujeme více záznamů z jiné tabulky. Tento vztah je nejpoužívanější typ relace.
- M:N – několika záznamům z jedné tabulky je možné přiřadit několik záznamů z tabulky druhé. Tento vztah bývá prakticky nejčastěji realizován kombinací dvou vztahů 1:N a 1:M,

které ukazují do pomocné, tzv. vazební tabulky složené z kombinace obou použitých klíčů. [6]

2.3.4 Objektově orientované databáze

Vedle relačních databází začali lidé vyvíjet nový typ databázových systémů. Teto databázový systém je založen na principech objektového programování. Základem objektově orientované databáze není již tabulka, ale objekt. Objektové databáze využívají datového modelu, který má objektově orientované aspekty jako třídy s atributy a metodami a integritními omezeními; poskytují objektové identifikátory pro každou trvalou instanci třídy (objekt). Podporují zapouzdření, násobnou dědičnost a podporují abstraktní datové typy. Objektově orientované databáze výrazně rozšiřují možnosti tvorby databázových aplikací. [5]

2.3.5 Objektově-relační databáze

"Rozšířená relační" a "objektově-relační" jsou synonyma pro databázové systémy, které se snaží sjednotit rysy jak relačních, tak objektových databází. Objektově-relační databáze využívají datový model tak, že do tabulek přidávají objektovost. Všechny trvalé data jsou neustále v tabulkách, ale některé položky mohou mít bohatší datovou strukturu, nazývanou abstraktní datové typy. Podpora abstraktních datových typů je výhodná, protože operace a funkce asociované s novými datovými typy mohou být použity k indexování, ukládání a získávání záznamů na základě obsahu nového datového typu. Do této kategorie patří např. Informix, IBM, Oracle a Unisys.

2.4 Datový sklad

Datový sklad (anglicky Data Warehouse, případně DWH) je zvláštní typ relační databáze, který umožňuje řešit úlohy zaměřené převážně na analytické dotazování nad rozsáhlými soubory dat. Tento typ databáze je určený primárně pro analýzy dat v rámci Business Intelligence.

Bill Inmon, který je považován za „otce“ datových skladů, definuje datový sklad takto: „Datový sklad je podnikový strukturovaný depozitář předmětově (subjektově) orientovaných, vzájemně provázaných, nepodléhající změnám, časově proměnných, historických dat používaných na získávání informací a pro podporu rozhodování. V datovém skladu jsou uložena detailní (atomická) i sumární data.“

2.4.1 Rozdíl mezi provozní relační databází a datovým skladem

U běžné relační databáze (**OLTP – On Line Transaction Processing**) je obvykle snaha o co nejmenší redundanci (opakování) uložených dat. Této co nejmenší redundance je dosahováno normalizací dat a vnitřním provázáním jednotlivých logických funkčních celků. OLTP popisuje zpracování dat v operativní databázi. Zaměřuje se na ukládání a zpracování záznamů o jednotlivých

uskutečněných transakcích a typicky zajišťují aktuální stav jisté evidence. Z hlediska časového jsou v systému OLTP významná aktuální data a data historická jsou archivována mimo OLTP systém, aby nedocházelo k jeho zbytečnému zatěžování. [7]
U **datového skladu** se vždy snažíme o jasnou vnitřní separaci jednotlivých funkčních celků. Výsledkem je tedy struktura, která je čitelnější pro uživatele, za ceny zvýšených nároků na prostor v paměti. Při popisu struktury datového skladu mluvíme o multidimenzionální (vícerozměrné) struktuře uložených dat. [8]
Z hlediska datového skladu slouží OLTP systémy jako zdroje dat, která jsou pravidelně čerpána pomocí „datové pumpy“.
Základním cílem datového skladu je uložit sjednocená a integrovaná data pro zefektivnění následující analytické a statistické práce s daty. Využívá přitom data historická.

2.4.2 Struktura datového skladu

Data v datovém skladu jsou z logického pohledu členěna do schéma (topologické uspořádání). Každé schéma odpovídá jedné analyzované funkční oblasti.

Schéma obsahuje dva typy tabulek – faktové a dimenzionální.

Jádro každého schématu tvoří jedna velká nebo několik faktových tabulek. V těchto tabulkách jsou vlastní analyzovaná data – veličiny, které sledujeme, hodnoty, které jsou používány k analytickým výpočtům (agregacím, třídění apod.) Většinu místa v paměti zabírají faktové tabulky, které obsahují detailní údaje ze všech zdrojů.

S faktovou tabulkou se pojí pojem granularita. Granularita určuje úroveň podrobnosti v tabulce faktů. Čím nižší je úroveň granularity, tím detailnější jsou data určená k provádění matematických operací.

Faktové tabulky jsou pomocí cizích klíčů spojeny s dimenzemi. Dimenze jsou tabulky, které obsahují seznam hodnot sloužících ke kategorizaci a třídění dat ve faktových tabulkách (atributy, prostřednictvím kterých se díváme na data). Je to vlastně číselník, podle kterého chceme data analyzovat.

Vlastnosti dimenzí:

- Dimenze určují úhel pohledu – čas, produkt, zákazník,...
- Dimenze určují hierarchie.
- Vztah mezi faktovou tabulkou a dimenzemi je 1:N [7][8]

2.5 Přehled nejčastěji používaných systémů řízení báze dat

2.5.1 PostgreSQL

PostgreSQL je relační databázový systém. Oplývá vynikající spolehlivostí a bezpečností. PostgreSQL je šířen pod BSD licencí, což umožňuje neomezené beplatné používání, modifikaci a distribuci PostgreSQL, a to jak pro nekomerční, tak komerční využití. Běží nativně na všech rozšířených operačních systémech včetne Linuxu, UNIXů a Windows. Za vývojem PostgreSQL stojí skupina nezávislých vývojářů, ke které se může kdokoli připojit. [9]

Výhody:

- open source (BSD licence)
- rozšiřitelnost – systém může být bezproblémově rozšiřován o nové datové typy, funkce operátory, agragační funkce, procedurální jazyky
- spolehlivost a bezpečnost
- vynikající uživatelská podpora

Nevýhody:

- slabší podpora uživatelského prostředí

2.5.2 Oracle

Oracle je systém řízení báze dat, který je vyvíjen společností Oracle Corporation. Oracle je moderní multiplatformový databázový systém s velmi pokročilými možnostmi zpracování dat, vysokým výkonem a jednoduchou strukturovatelností. Patří mezi relační databázové systémy, ale podporuje také objektové databáze a hierarchický model uložení dat. [10]

Výhody:

- jeden z vedoucích produktů na IT scéně
- výborná podpora
- využití i pro velké objemy dat

Nevýhody:

- cena
- closed source a z toho vycházející neprůhlednost

2.5.3 Informix

Výhody:

- jeden z vedoucích produktů na IT scéně
- výborná podpora

Nevýhody:

- cena
- closed source a z toho vyplývající neprůhlednost

2.5.4 MySQL

MySQL je databázový systém, vytvořený švédskou firmou MySQL AB, původně vlastněný společností Sun Microsystems, dceřinou společností Oracle Corporation. MySQL bylo již od počátku optimalizováno na rychlost i za cenu některých zjednodušení. Tyto vlastnosti jsou však v posledních letech doplňovány.

Mezi typická použití databáze MySQL můžeme zařadit její nasazení na Linuxové internetové servery spolu s Apache a s PHP, nazýváno též technologie LAMP. [11]

Výhody:

- rychlost
- nejrozšířenější databáze
- open source (GPL)
- dobré uživatelské prostředí

Nevýhody:

- nemá tolik specializovaných funkcí jako jiné konkurenční systémy

V roce 2010 byla ochranná známka MySQL spolu s právy na rozšiřování produktu zakoupena firmou Oracle Corporation, a ta dále vyvíjí některé grafické prvky a i některé porty na další platformy. Oracle Corporation ale nezohledňuje dostatečně požadavky komunity na vývoj, a tak komunita pokračuje ve vývoji tohoto produktu pod jménem MariaDB.

2.6 SQL

SQL (Structured Query Language – Strukturovaný dotazovací jazyk) je jazyk sloužící pro tvorbu dotazů v databázích. [12] V praxi se však nejedná o jeden jazyk, ale o různé dialekty šité na míru různým softwarovým produktům, které se více či méně podřizují SQL standartu. Na základní úrovni ho však můžeme považovat za jeden jazyk. [13] Zahrnuje v sobě nástroje pro jak tvorbu struktur

databází, tak nástroje pro manipulaci s daty. SQL patří mezi deklarativní programovací jazyky. Deklarativní jazyk je založen na myšlence programování aplikací pomocí definic toho, co se má udělat a ne jak se to má udělat.

Jazyk SQL se skládá z několika částí:

- Data Definition Language – jazyk pro vytváření databázových schémat a katalogů
- Data Manipulation Language
- View Definition Language – jazyk určující vytváření pohledů
- Storage Definition Language – definuje způsob ukládání dat do tabulek [5]

3. Anonymizace dat

3.1 Osobní údaj

Podle ustanovení § 4 písm. a) zákona č. 101/2000 Sb. se osobním údajem rozumí jakákoliv informace týkající se určeného nebo určitelného subjektu údajů. Každá informace může být osobním údajem, pokud vede k identifikaci konkrétního člověka, byť třeba v kombinaci s dalšími informacemi. [14]

3.2 Identifikační údaj

Jsou to takové údaje, které jednoznačně nebo téměř jednoznačně odlišují konkrétní jedince:

- jméno, příjmení
- rodné číslo (gatantovaná jednoznačnost v rámci České republiky)
- adresa
- telefonní číslo atd.

3.3 Citlivý údaj

Citlivým údajem se rozumí osobní údaj vypovídající o národnostním, rasovém nebo etnickém původu, politických postojích, členství v odborových organizacích, náboženství a filozofickém přesvědčení, odsouzení za trestný čin, zdravotním stavu a sexuálním životě subjektu údajů a genetický údaj subjektu údajů; citlivým údajem je také biometrický údaj, který umožňuje přímou identifikaci nebo autentizaci subjektu údajů. [15]

3.4 Anonymní údaj

Anonymním údajem je takový údaj, který buď v původním tvaru nebo po provedeném zpracování nelze vztáhnout k určenému nebo určitelnému subjektu údajů. [15]

3.5 Anonymizační techniky

Anonymizovaná data jsou taková, která ani nepřímo nepomáhají v identifikaci určitého člověka a nejsou s ním tedy nijak spojitelná.
Kvalita anonymizace dat je měřena mírou obtížnosti identifikace subjektu – tedy spojením konkrétní osoby s konktrétním záznamem s jistotou nebo s vysokou pravděpodobností.

3.5.1 Výmaz

Výmaz můžeme definovat jako přepsání konstantní, většinou prázdnou, hodnotou. Výjimečně, např.

pro zachování některých statistických vlastností, používáme neprázdnou, ale generickou hodnotu (např. aritmetický průměr). Použitelné u statisticky nesledovaných vlastností záznamu (např. jméno a přijmení)

Výhody:

- zaručené odebrání identifikačního prvku

Nevýhody:

- potenciální ztráta integrity databáze

3.5.2 Hash

Jde o funkci, která převádí vstupní posloupnost bitů na posloupnost pevné délky n bitů.

Výhody:

- zachování integrity databáze
- zaručené odebrání zpětné identifikace při dostatečně velkém počtu možných vzorků

Nevýhody:

- při malém počtu vzorků jde data získat útokem brutální silou (brute force)
- možnost vzniku kolize – kolize nastává, když dvojice různých vstupních posloupností má stejný otisk (stejnou výstupní posloupnost bitů). U kvalitní hash funkce je pro vstupní poslopnosti do n-1 bitů tato možnost velmi malá. Pro vstupní posloupnosti nad n bitů je sice tato možnost matematicky nevyhnutelná, ale klíčovou vlastností dobré hash funkce je praktická nechopnost najít vstupní posloupnost odpovídající konkrétní výstupní posloupnosti a tím nalézt kolizní pár účelově.

Message-Digest algorithm (MD5) – Algoritmus MD5 se prosadil do mnoha aplikací (např. kontrola integrity hesel nebo ukládání hesel). MD5 je popsán v internetovém standardu RF C1321 a vytváří otisk o velikosti 128 bitů. Byl vytvořen v roce 1991 Ronaldem Rivestem a nahradil dřívější hashovací funkci MD4. Algoritmus MD5 byl překonán teoreticky i prakticky. Od roku 2004 je znám algoritmus, který umožňuje nalezení kolizního páru. [16]

Secure Hash Algorithm (SHA)

- SHA-1 – tvoří obraz zprávy dlouhý 160 bitů. SHA-1 byl překonán teoreticky (od roku 2005), ale zatím nepřekonán prakticky. Zatím neexistují výpočetní prostředky pro dosažení praktického prolomení.
- SHA-2 – Do skupiny SHA-2 patří algoritmy SHA-224, SHA-384 a SHA-512. Čísla v těchto názvech značí délku výstupního otisku v bitech. Zatím nebylo SHA-2 překonáno ani

teoreticky, ani prakticky.

3.5.3 Perfektní klíč (perfect key)

Jedná se o dosazení náhodně vygenerovaných unikátních hodnot, a tedy vytvoření 1:1 asociace mezi původním neanonymním údajem a nově náhodně vygenerovaným, tedy již anonymním údajem. Uchováme-li tuto asociaci pro pozdější použití, můžeme výsledky práce s anonymizovanými daty pak jednoduše aplikovat na původní data pro získání specifických neanonymních výsledků.

3.5.4 Kategorizace

Při znalosti nejmenšího vyhotoveného rozlišení konkrétního údaje záznamu je možno proměnit proměnnou s vysokým rozlišením v proměnnou označující jeden z rozsahů (rozlišení nižšího stupně) – tedy kategorizovat a v případě numericky vyznačených plynulých proměnných “zaokrouhlit”; Takto jsou zachovány aplikačně (statisticky) důležité údaje, zatímco jsou odstraněny identifikační prvky; Např. máme-li dárce krve, u kterých měříme jejich výšku v milimetrech a váhu v gramech, pak zaokrouhlením výšky a váhy na cm a dkg nebo dokonce na dm a kg můžeme v databázi zachovat potřebné údaje pro vědecké závěry, zatímco nuance, které by identifikovaly konkrétní osoby, tímto vymizí.

4. Data mining

Data mining (nebo také dolování/vytěžování dat) je soubor metod sloužící ke zpracování dat a k získání ryzí formy informací v nich obsažených. Data mining je matematická metoda, jak získávat skryté, ale užitečné informace z obrovského množství údajů. Spadá pod obor business intelligence. Dobývání dat se používá v mnoha oblastech a nastoupilo s rozvojem počítačové techniky, která zpracování obrovských databází umožnila. Nejčastěji se však používá v oblasti marketingu. [17]

4.1 Historie data miningu

První náznaky data miningu se objevily v 60. letech 20. století s rozvojem počítačové techniky. Šlo například o regresní analýzy s automatickým výběrem proměnných a prvních rozhodovacích stromů. Většinou však šlo jen o ojedinělé nebo akademické záležitosti. Rozvoj statistických metod, databázových aplikací a umělé inteligence spolu s rychlým růstem rychlosti a paměti počítačů byly předpoklady, které umožnily v sedmdesátých a osmdesátých letech první systematická využití data miningové metodologie v praxi.

Slovní spojení data mining tehdy ovšem stále mělo spíše hanlivý přídech: Označovalo vybírání z dat, hledání korelací ve velkých datových souborech, které je vystaveno obrovskému nebezpečí, že objeví pouze nahodilé fluktuace v datech bez možnosti zobecnění a praktického využití.

Obrat přišel počátkem devadesátých let. V té době byly již vybudovány metody, umožňující vyhnout se zmíněnému nebezpečí falešných korelací. Navíc zejména v USA rostla poptávka ze strany komerčních organizací, disponujících již velkými objemy dat a neschopných z nich pomocí klasických metod získat potřebné podklady pro rozhodování. To napomohlo k rychlému etablování data miningu jako svébytného oboru aplikované vědy a k jeho širokému použití v komerční praxi. Časté aplikace jsou především v oblastech přímého marketingu (výběr klientů pro oslovení), finančnictví (např. odhadování rizika, hledání podvodů), maloobchodního prodeje (analýza nákupních košíků aj.), telekomunikací (segmentace klientů, prodej programů aj.) a internetového prodeje (analýza přechodů mezi stránkami, efektivity reklamy apod.).

Nárůst aplikací v oblasti data miningu se projevil i na softwarovém a konzultačním trhu. Existuje již poměrně široká nabídka specializovaných softwarů pro tento účel. Vedoucími trhu datamininingových softwarů jsou komerční aplikace SAS Enterprise Miner, SPSS Clementine a STATISTICA Data Miner, mezi známé nekomerční softwary patří Weka a Orange.

4.2 Cross-Industry Standard Proces for Data Mining

Různorodost počátků tohoto oboru vedla k zavedení standardizovaného metodologického postupu. Tento postup byl pojmenován CRoss-Industry Standard Proces for Data Mining (CRISP-DM) a jak

můžeme z názvu vyčíst, jedná se o standardizovaný proces pro všechny obory, je tedy jedno, odkud analyzovaná data pocházejí. [17]

Metodologie popisuje data mining v následujících šesti krocích:

1. Pochopení problému – v prvním kroku je třeba porozumět požadavkům zákazníka a stanovit si jasný cíl. V této fázi dochází také k návrhu a tvorbě plánu pro řešení daného problému.
2. Porozumění datům – další krok nezbytný pro další vývoj procesu. V tomto kroku dochází také k vytvoření hypotéz, které se snažíme v průběhu celého procesu data miningu potvrdit nebo vyvrátit.
3. Příprava dat – zde dochází k integraci více datových zdrojů, čištění a úpravě dat do podoby, kterou vyžadují analytické nástroje a metody, které později budou na data aplikovány. Tento proces nelze správně provést bez znalosti dat. Špatná integrace dat může vést ke znehodnocení zdrojů dat a ovlivnění celkové kvality řešení.
4. Modelování – tato fáze procesování dat obsahuje testování vhodných metod a nastavení jejich parametrů pro řešení definovaného problému. Z tohoto kroku vybíráme několik nejlepších získaných řešení, které postupují do dalšího kroku.
5. Hodnocení – v předposlední fázi dochází ke konečnému hodnocení a selekci získaných modelů podle různých vlastností a ověření správnosti získaných řešení za pomoci těchto modelů. Dle získaných výsledků je již možno zvážit případnou implementaci celého procesu.
6. Nasazení – nasazení je posledním krokem v celém procesu. Musíme však podotknout, že v tomto bodě proces nekončí, ale začíná se cyklicky opakovat. Pokud se zákazník rozhodne výsledky data miningu implementovat do svých procesů, je nezbytné modely udržovat aktuální. Závislosti v datech se časem mění, a pokud by systém nebyl dostatečně robustní či pravidelně aktualizován, je velmi pravděpodobné, že by časem pozbyl kvality, tak i zcela své funkce. Proto je nutné pravidelně ověřovat funkci modelu novými daty a tím udržovat aktuálnost modelů. [17]

Obecně zde navíc platí víc než jinde pravidlo Garbage In Garbage Out (GIGO), čili smetí dovnitř smetí ven. Z nekvalitních zdrojů nikdy nedostaneme kvalitní informace.

4.3 Současné nástroje pro data mining

Orange

- open source software (GNU General Public License)
- počátek vývoje v roce 1993 na University of Ljubljana
- jak vizuální programování, tak skriptování v jazyce Python

- jednoduché uživatelské prostředí

KNIME (the Konstanz Information Miner)

- opensource (GPLv3)
- odvozen od platformy Eclipse a d9ky tomu mimořádná modulárnost produktu
- hodnotná rozšíření především v oblasti biochemie, KMINE se těší mimořádné popularitě ve výzkumu ve farmaceutickém průmyslu
- vzhledem k platformě Eclipse je nejpřirozenější volbou pro vytváření rozšíření jazyk Java, ale použít lze i Python a další jazyky [18]

SAS Enterprise Miner

- jeden z modulů systému SAS
- SAS byl v letech 1966 – 1976 vyvýjen na North Carolina State University
- v roce 1976 vznikla korporace SAS Institute, kde pokračoval jeho další vývoj
- využívá jak vizuální programování, tak svůj vlastní programovací jazyk SAS programming language

STATISTICA Data Miner

- nástroj pro vytěžování dat založený na velmi jednoduchém uživatelském prostředí
- rozsáhlý výběr analytických technik (široký výběr algoritmů na shlukování, pro různé typy neuronových sítí, interaktivních regresních a klasifikačních stromů, vícerozměrného modelování a mnoho dalších)
- optimalizováno pro zpracování extrémně velkých dat i z více zdrojů simultánně [19]

Weka (Waikato Environment for Knowledge Analysis)

- program strojového učení napsaný v Javě vyvinutý na University of Waikato na Novém Zélandě
- opensource program s GNU General Public License
- obsahuje sbírku nástrojů pro vizualizaci, datovou analýzu a prediktivní modelování
- vyvýjen od roku 1993, v roce 1997 byl reimplementován v Javě

5.Grafické uživatelské prostředí

Grafické uživatelské prostředí (GUI) je takové rozhraní, kde jednotlové části, kterým říkáme objekty, jsou reprezentovány graficky (nebo např. textově, zvukově či jinak) a uživatel může s těmito objekty manipulovat pomocí ukazovacího zařízení (myš, klávesnice, dotyková obrazovka,...). Činnosti, které lze s objekty vykonávat nazýváme akce.

5.1 Pravidla pro návrh uživatelského prostředí

1. Konzistence – při tvorbě uživatelského prostředí je vhodné směřovat k tvorbě stereotypů. Výsledkem by mělo být, že „stejné věci se dělají stejně, podobné věci se dělají podobně“.
2. Respekt široké škály uživatelů – autor aplikace by si měl předem ujasnit, kdo jsou cíloví uživatelé (profesionálové, děti, začátečníci,...) a podle toho by měl danou aplikaci tvořit. Různí uživatelé mají různý způsob práce (např. začátečnící preferují vyvolání akce z panelu nástrojů, středně pokročilí z menu a pokročilí pomocí klávesové zkratky), proto je důležité, aby existovaly alternativy v ovládání aplikace. Prostředí se vždy přizpůsobuje uživateli, ne uživatel aplikaci. [20]
3. Zpětná vazba – uživatel potřebuje být přiměřeně informován o výsledcích práce aplikace. Je však třeba volit formu tak, aby aplikace uživatele neobtěžovala. Rozlišujeme silnou a slabou zpětnou vazbu. Silná vazba je taková, kdy uživatel musí dát najevo, že vzal zprávu na vědomí nebo na ní musí přímo reagovat některou z nabízených možností. Slabá zpětná vazba je taková, kde uživatel nemusí potvrzovat, že se s informací seznámil. Silná vazba je vhodná pro informace, které mají pro uživatele zásadní význam. Slabá zpětná vazba je vhodná pro vedlejší nebo doplňující informace.
4. Navigace uživatele – řada uživatelských úloh se zkládá z posloupnosti více akcí. Je vhodné rozdělit rozhraní aplikace na jednoduché, logicky rozčleněné kroky.
5. Předcházení chyb – uživatelské rozhraní by mělo minimalizovat možné chyby uživatele. V případě, že k chybě dojde, je třeba uživate o ní informovat. Je také dobré zdělit uživateli možné příčiny problému a navrhnout možná řešení. Také je třeba volit vhodný způsob a jazyk, kterým budeme informovat. Je vhodné se vyhnout spoustě technických informací, které jsou pro uživatele bezcenné.
6. Tolerance k chybám uživatele – je vhodné, aby mohl uživatel všude tam, kde je to možné provést akci zpět. Pokud vyvolá akci, musí být schopen ji se z ní vrátit nebo ji alespoň pozastavit. Možnost se uživateli vrátit je podstatný rozdíl mezi uživatelským prostředním a znakově orientovaným rozhraním (např. příkazová řádka), kde není možnost vzít svou akci zpět.
7. Předvídatelnost – uživatelské prostředí by mělo být předvídatelné. Uživatel by měl být

řídícím prvkem rozhraní, měl by být iniciátorem, nikoliv tím, kdo plní rozkazy aplikace.

8. Nepřetěžování paměti uživatele – uživatel nesmí být nucen si rozhraní pamatovat. Je potřeba, aby měl přehled o aplikaci, bez nutnosti si jednotlivé věci pamatovat nebo je znovu a znovu číst. Platí psychologické pravidlo 7±2. Toto pravidlo říká, že člověk si je schopen v krátkodobé paměti uchovat pět až devět údajů. [21]

5.2 Vizuální uspořádání prvků

Při rozmisťování ovládacích prvků v dialozích (oknech) musíme brát v úvahu dvě základní pravidla:

- uživatel prohlíží obrazovku od levého horního rohu po směru hodinových ručiček
- v našem prostředí se čte zleva-doprava a shora-dolů. Úkolem je tedy rozmístit ovládací prvky tak, abychom minimalizovali dráhu očí nutnou pro zpracování obsahu okna.

Když vezmeme v úvahu tato pravidla, budeme nejdůležitější ovládací prvky umisťovat od levého horního rohu. Při rozmisťování dalších ovládacích prvků je otázkou, zda upřednostnit směr zleva-doprava nebo zhora-dolů.Výzkumy ukazují, že vhodnější je upřednostnit směr zhora-dolů. Dráha oka potřebná pro zpracování takové obrazovky je při tomto uspořádání kratší a efektivnější. Jedinou výjimkou, kdy je vhodné upřednostnit směr zprava-doleva před směrem shora-dolů je u oken, v nichž převažuje informace ve formě souvislého textu. [20] [21]

Dále se podíváme na několik zásad pro vizuální organizaci ovládacích prvků.

1. Zásada vyváženosti – okno by mělo být zaplněno ovládacími prvky jak horizontálně, tak vertikálně.
2. Zásada souměrnosti – členění skupin ovládacích prvků ma být stejné nebo podobné jak na levé, tak na pravé straně okna.
3. Zásada pravidelnosti – rozměry stejných ovládacích prvků, jejich barva, tvar, umístění, by měly být všude tam, kde je to vhodné, jednotné.
4. Zásada předvídatelnosti – styl a logika uspořádání prvků v oknech a jejich ovládání by mělo být totožné ve všech oknech.
5. Zásada následnosti – ovládací prvky mají být rozmístěny podle logiku problému zhora-dolů, zleva-doprava. Následnost může být také podpořena barvami. Lidské oko preferuje světlejší barvy před tmavšími, barevné před černobílými, syté barvy před pastelovými. Následnost můžeme také podpořit seskupením – oko preferuje oddělené ovládací prvky před seskupenými, prezentací – oko preferuje symbol před textem, rozměry – oko preferuje velké

objekty před malými nebo tvarem – oko preferuje jednoduché a oblé tvary.

6. Zásada účelnosti – vždy se řídíme pravidlem, že forma sleduje funkci. Používáme jednoduchý styl, střídmě nakládáme s barvami, tvary apod. Uživatelské prostředí musí být funkční, jedině tehdy může být i pěkné.
7. Zásada jednotnosti – ovládací prvky v okně by měly tvořit jeden vizuální celek. Pro ovládací prvky stejného nebo podobného významu používáme stejné rozměry, tvar, barvu apod.
8. Zásada proporce – při rozmisťování prvků v okně klademe důraz na celek, soulad jednotlivých částí, rozměry. Existuje několik poměrů stran, které jsou obecně považovány za estetické:
   - čtverec (1:1) – tento základní tvar přitahuje pozornost uživatele
   - zlatý řez (1: $\frac{1+\sqrt{5}}{2}$) - poměr, které ma velkou tradici ve výtvarném umění a architektuře
   - odmocnina ze tří (1: $\sqrt{3}$)
   - mužeme také použít poměry jako 1:2, 1:3, ale v dnešním světe jsou pokládány za umělé a technické
9. Zásada jednoduchosti – okno by mělo obsahovat přiměřený počet prvků tak, aby byla zachována jednoduchost a přehlednost.
10. Zásada seskupování – pro zachování přehlednosti je vhodné prvky logicky seskupovat podle toho, jaký mají význam a účel. [21]

PRAKTICKÁ ČÁST

Pro konkrétní přínos této diplomové práce jsem spolupracovala s Krevním centrem Fakultní Nemocnice Ostrava (KC FNO), kde mi bylo umožněno aplikovat znalosti do praxe za účelem přínosu konkrétního využitelného nástroje výzkumné a částečně možná i každodenní práce Krevního centra.

Studie a poznatky teoretické části této diplomové práce se odráží ve výsledcích její praktické části, i když pro dosažení praktické hodnoty pro Krevní centrum bylo zapotřebí přistoupit ke kompromisům, a tak některé aspekty teoretické části byly zvýrazněny více (práce s relačními databázemi) a některé méně (vlastní design GUI elementu).

Krevní centrum za dlouhou dobu svého trvání nashromáždilo velké množství dat, které v sobě ukrývají jak indikátory možných vylepšení procesů Krevního centra, tak důležité medicínské podklady pro studie onemocnění.

Tyto důležité podklady jsou nashromážděny ve dvou hlavních databázích. V původní databázi ELAS a v současné operační databázi. Pro účely této diplomové práce byla vybrána původní databáze ELAS.

Prvním krokem, který jsem v rámci vypracování praktické části diplomové práce provedla, byla anonymizace databáze Krevního centra FNO. Tento klíčový krok bylo nutné provést hned na počátku z důvodu ochrany soukromí pacientů a dárců KC FNO.

V druhé fázi pak bylo třeba převést takto získaná data do databáze jiné, nám přístupné. V tomto případě připadalo v úvahu několik opensource řešení, jako SQLite, PostgreSQL a MySQL/MariaDB. Pro krevní centrum je však vhodnější centrální systém řízení báze dat, a tak jsem tedy SQLite vyloučila. Pro lepší podporu uživatelského prostředí pak nakonec zvítězila MySQL/MariaDB nad PostgreSQL.

Ve třetí fázi jsem se zaměřila na dodání grafického nástroje pro přístup k těmto údajům, na jejichž podkladě pak může být vypracovávána další statistická analýza a jiné závěry.

Z výsledků a závěrů praktické části této diplomové práce pro Krevní centrum dále vyplývá, že ač je výsledný nástroj této diplomové práce pro některé účely již zcela použitelný, tak pro specializovanější použití, jako například pro více komplexní dotazy na databázi nebo slučování s externími zdroji dat, bude nezbytné tuto databázi nejprve příslušnými prvky optimalizovat,

případně zvýšit kvalitu dat. Například pro výzkum závislosti výsledků krevních vyšetření na historických hladinách smogu by bylo vhodné doplnit chybějící PSČ, které se v datavázi nenachází u cca 85 % registrovaných osob, na základě města jejich bydliště v původní, neanonymizované verzi ELASu. Následně pak provést kapacitní rozvržení hardwaru na straně serveru podle počtu současných uživatelů a složitosti jejich dotazů. Na straně uživatelského software není potřeba žádných význačných optimalizací nebo kapacitního rozvržení.

Při vypracování této diplomové práce bylo téměř zcela použito technologií přístupných veřejnosti bez licenčních poplatků, tedy vesměs opensource technologie s licencí GNU GPL.

6. Popis databáze ELAS a její anonymizace

Zde je nutno podotknout, že ELAS je cca 20 let starý systém a že dokumentace je přístupná pouze v papírové formě. Není tedy možno použít počítače pro dohledávání vzájemných vztahů a co hůř, identifikátorů přiřazených k popisům a významům. Práci s dokumentací ELASu taktéž stěžuje skutečnost, že názvosloví a identifikátory jsou velmi úsporné (tedy kryptické), nepopisující intuitivně význam datových elementů, které označují. Např. v jedné tabulce je sloupec nazván „cd“ a v jiné sloupec s aplikačně nutně stejnými hodnotami je nazván „cvys“, v jedné „z_diag“ a v jiné „kod“. Hledání vzájemného propojení databázových klíčů na úrovni aplikačních pravidel bylo vskutku detektivním úkolem. Považíme-li, že přístup k této databázi byl skrze database dump o více jak 3500 souborech (co soubor, to tabulka), z nichž bylo vybráno více než 530 jako přímo relevantní k operačnímu účelu Krevního centra a z těchto 530 tabulek pak došlo k zúžení na 474 tabulek použitelných pro účely této diplomové práce. Z tohoto vyplývá, že v praktické části bylo zapotřebí si i z důvodu množství námahy vybírat, které komponenty výsledného řešení bylo možno v rámci této diplomové práce postavit od základu a pro které bylo zapotřebí užít či adaptovat existující technologie tak, aby tato diplomová práce byla použitelná v praxi.

Klíčové bylo rozhodnutí, zda-li je pro využití pro vědecké účely zapotřebí zamezit re-identifikaci nebo je naopak zapotřebí případné dohledání jedinců pro jejich ochranu nebo ochranu společnosti. Problematika, po poradě se zaměstnanci Krevního centra, byla vyřešena skrze první přístup, který považuji z technických důvodů za bezpečnější pro ochranu osobních údajů.

6.1 Určování identifikujících informací

Jedním z hlavních kriterií předmětné anonymizace je zachování statisticky významných prvků pro účely dalšího zpracování a data-miningu dosavadních dat. Bylo tedy třeba označit údaje, které by mohly být použity pro identifikaci a odloučit z nich takovou informační složku, aby výsledná data již pro identifikaci osob použitelná nebyla a zároveň aby byly zachovány statisticky významné složky těchto údajů. Z těchto údajů byly ve spojení s pracovníky Krevního centra shledány identifikující údaje jméno, příjmení, rodné číslo, adresa, telefonní číslo a e-mailová adresa jako identifikující prvky s tím, že pohlaví a datum narození osob nejsou dostatečně identifikujícími položkami pro jejich vyloučení a tak byly zachovány.

Anonymizace spočívala v odstranění jmen, příjmení, adres a telefonních čísel osob nalézajících se v dané databázi.

Protože rodná čísla jsou v ELAS databázi používána jako klíče relačních vztahů, byla tato čísla

zaměněna v hodnoty jiné (anonymizované). Vzhledem k důležitosti informace o pohlaví a věku pacienta či dárce, jsme se rozhodli zachovat formát anonymizovaného rodného čísla RRPMDD/XXX[Y]. Abychom zdůraznili, že se zde nejedná o platná rodná čísla, tak jsme zvolili nepoužít desátou Y číslici jako kontrolní číslici (u platných rodných čísel se desátá číslice dopočítává jako kontrolní tak, aby výsledné číslo bylo beze zbytku dělitelné jedenácti). Naše anonymizované „rodné číslo“ je tedy ve formátu RRPMDD/ZZZ. Další možností, jaký formát anonymizovaného čísla zvolit, by bylo dopočítání desáté číslice tak, aby výsledné číslo nebylo dělitelné jedenácti, a tím každému uživateli nebo systému, který s výslednými daty přijde do styku a je ochoten si nejprve dělitelnost ověřit, zaručili, že se o platné rodné číslo nejedná. Před rokem 1954 byla devítimístná rodná čísla bez kontrolní desáté číslice běžně vydávána a byla platná.

V polích rodných čísel tedy byly zachovány informace o pohlaví a datu narození, ale identifikující část platného rodného čísla, tedy poslední tři číslice spolu s čtvrtou kontrolní, byla předmětem anonymizace. Protože jsme se jak pro náročnost, tak i pro zbytečnost, chtěli vyhnout změně databázového schématu, zbývalo tak určit, jak poslední čtyři pozice rodného čísla zaměnit. Výběr příslušné anonymizační metody pro tento údaj byl snadný. Zaprvé je key space o tisíci vzorcích (nebo i o 36 miliónech možností dat narození za posledních sto let) extrémně malý pro zajištění bezpečnosti skrze hash funkci. Zadruhé žádnou hash funkci, která by v rámci tří (popřípadě čtyř) takto uvolněných znaků jakkoli rozumně garantovala bezkolizní mapování, jsme neměli k dispozici. Způsob kategorizace by byl taktéž neuplatitelný, protože se nejedná o spojitou proměnnou a nepotřebujeme zachovávat statistické informace. Jako jasný kandidát vyvstalo použití perfektního klíče.

Tento perfektní klíč byl vygenerován sice pseudo-náhodně, tedy nikoli zcela náhodně, ale generujícím klíčem pro toto vygenerování byl obsah samotné původní databáze. Tedy k tomu, aby byl někdo schopen prolomit takto vygenerovaný perfektní klíč, musel by mít k dispozici samotnou původní neanonymizovanou databázi.

Abychom mohli tento perfektní klíč vytvořit, bylo nejprve zapotřebí projít všemi tabulkami databáze, abychom nalezli všechny výskyty sloupců, které by rodná čísla vůbec mohly obsahovat. Tímto se vyloučily sloupce o datových typech jako date, time, datetime, a decimal, char nebo varchar kratší než 10 znaků. ELAS však byl bezpečně přístupný pouze jako database dump, tedy export přímo ze systému a zcela chyběla elektronická dokumentace významů polí a jejich provázanosti, takže se u takového dumpu o počtu cca 500 tabulek muselo nejprve nelehce zjistit, ve kterých polích se rodná čísla vyskytují. Celkem je v 474 tabulkách více jak 4800 sloupců databáze a v nich se našlo více než 1600 sloupců, které by mohly nést informaci o rodném čísle. Prozkoumávání takového počtu sloupců a vyhledávání citlivých údajů, byla sice mravenčí práce, ale pro postup vpřed s diplomovou prací ve směru praktické použitelnosti jejich výsledků pro KC FNO, bylo dotáhnutí tohoto kroku nezbytné.

Nakonec bylo identifikováno cca 75 sloupců v 50 tabulkách, které bylo zapotřebí anonymizovat, a to navíc tak, aby nebyla porušena integrita databáze.

Pak následovala další mravenčí práce při prohledávání těchto sloupců a nalézání těch, které nějaké devíti a deseti-místná čísla obsahovaly, případně v typu char a varchar s lomítkem za prvními šesti znaky. Problém byl ale ten, že se nedá spoléhat na to, že každý řádek, každého z více než 1600 kandidujících sloupců by, kdyby byl nositelem rodných čísel, obsahoval rodné číslo. Protože z tohoto pohledu je v ELASu kvalita dat poměrně nízká (např. jen u cca 15 % všech evidovaných dárců je zachyceno jejich PSČ), byla tím práce ještě o něco komplikovanější. Následující kód pomohl najít sloupce, které obsahovaly údaje, které mohly být rodnými čísly:

```
for i in /tmp/elas-db-dump/*unl.gz ;
 do echo -ne $i : '' ;
  zcat $i | perl -F '|'  -ane '
    while(defined $F[0 + $f++]) {
      $possible{$f-1}++ if $f ~= m:[0-9]{6}/?[0-9]{3,4}: };
   END{ print join("\t", ( keys @possible ), $/ ) }' ;
 done | less
```

Většina polí obsahující rodné číslo měla opravdu formát NNNNNN/NNNN a byla vyjádřena jako „char(11)“ typ. Našlo se ale i několik polí, pojmenovaných „rod_cis“, které mělo typ „char(10)“.

Naštěstí takto definovaná pole jsou buď ve všech řádcích zcela prázdná nebo, jak se později ukázalo, tato pole nebyla pro tuto práci potřebná či významná a nebyla tedy zahrnuta do výsledné práce.

Ve výsledku jsem našla něco přes čtvrt miliónu různých rodných čísel. Tímto jsme získali poměr počtu anonymizovaných rodných čísel ke key space, tedy k velikosti prostoru, do kterého tyto nově vygenerované údaje zakódovat. Uvážíme-li, že pro zakódování máme k dispozici prostor jednoho tisíce klíčů (jeden tisíc reprezentovaný třemi číslicemi za lomítkem rodného čísla) pro každé datum od roku cca 1915 do 2015, tak nám vychází, že bychom takto mohli zakódovat až 365 x 100 x 1000 = 36.500.000 klíčů. Ale potřebujeme zakódovat méně než setinu. Pro zvýšení entropie (pro rozšíření nutného hledání případnému útočníku na náš perfektní klíč), tedy můžeme trojčíslí za lomítkem vztáhnou nejen ke konkrétnímu datu vyjádřenému v šestici před lomítkem, ale dokonce jen k měsíci vyjádřenému v prvních čtyřech znacích (dva znaky rok + dva znaky měsíc + pohlaví). Zde pouze hrozí, že z důvodu jakési anomálie v porodnosti v některý měsíc za posledních 100 let nebo z potřeby v takovém měsíci narozených lidí být dárci nebo pacienty v Krevním centru FNO, by byl počet rodných čísel nad 1000. Pak by nebylo možno uplatnit jednoduché rozdělení klíče takto vůči

měsíci. Před převodem ELASu z dumpu z Informixu do MySQL tato verifikace byla poměrně těžší. Po převodu můžeme toto zkontrolovat poměrně lehce:

```
SELECT
   substr(rod_cis, 1, 4) as ym
   ,count(*) as pocet
 FROM p_ev
  GROUP BY  ym
   ORDER BY   pocet DESC
      LIMIT 3 ;
+------+-------+
| ym   | pocet |
+------+-------+
| 7455 |   360 |
| 7453 |   340 |
| 7454 |   336 |
+------+-------+
3 rows in set (0.22 sec)
```

Bylo potvrzeno, že více než 1000 pacientů ve stejném měsící narození nenastalo. Nejvyšší registrace v Krevním centru FNO byla zjištěna pro ženy v počtech 360, 336 a 340, a to narozených v měsících březen, duben a květen roku 1974. Pro zajímavost, ve 144 nejčastěji zastoupených měsících vedou, co do počtu pacientů, ženy (v počtech 360 – 184 pacientek na měsíc). Pro data narození v březnu 1943 má ELAS registrovaných více mužů než žen (183 pacientů). Myslím, že již jen při odhalení další úrovně rozdělení těchto dat o registraci dojde k zajímavým a možná i významným zjištěním jak pro běh Krevního centra, tak pro analýzu onemocnění v místní společnosti v závislosti na demografice.

7 Převod dat z Informixu do MySQL

„The reason I love standards is because there are so many to choose from.“
„Důvod proč miluji standardy je proto, že jich jsou na výběr takové spousty.“

Neznámý autor

V této kapitole se budeme zabývat převodem dat databáze ELAS z database dumpu vygenerovaného z databáze Informix do MySQL.

Převod schématu i dat z database dumpu Informixu do MySQL měl svá úskalí.

Ačkoli SQL má být standard, tak každý výrobce systému řízení báze dat si k němu vytváří vlastní přídavky, aby svým uživatelům dal maximální komfort pro využití výhod právě jeho produktu. Tím ale vznikají nekompatibilní formy SQL, které se pak naneštěstí nedají ani rychle a už vůbec ne jednoduše přetvořit jeden v druhý. Naštěstí naše data pro warehouse/data mining užití původně operační databáze ELASu nevyžaduje pro strohou funkčnost mnohá z optimalizací použitých v OLTP módu ELASu a tak jsem byla schopna použít, po odstranění většiny přídavků specifických pro Informix, DDL dump databáze ELAS.

Když však dojde na optimalizaci dotazů, a tak na optimalizaci přístupových a úložných struktur v MySQL, pak již bude zapotřebí sáhnout i po přídavcích MySQL produktu.

Mírné problémy nastaly při převodu kódování znaků, kdy data v databázi ELAS byla exportována v kódování „latin2“, ale výchozí nastavení MySQL pracuje s kódováním „latin1“. Po několika neúspěšných pokusech o přehození MySQL nastavení kontextu kódování v rámci databaze spojení jsem rezignovala, zkonvertovala všechna data z databáze ELAS z kódování latin1 do utf8, na pevno v konfiguračním souboru MySQL server procesu nastavila kódování utf8, restartovala MySQL server a načetla do MySQL obsah tabulek z utf8 verze ELASu :

```
for tablefile in /tmp/komplet-utf8/pozva00135.unl ;
do echo „LOAD DATA LOCAL INFILE '$tablefile'
 INTO TABLE  pozvanky
  CHARACTER SET utf8
    FIELDS TERMINATED BY '|';“ ;
done | mysql –u root –ppwd  isto_ov
```

Jenže Informix exportoval sloupce z databáze ELAS s typem „date“ a „datetime“ ve formátu DD.MM.RRRR a MySQL očekává formát data ve formátu RRRR-MM-DD. Nepovedlo se mi přesvědčit MySQL, aby očekávala jiný formát. Bylo tedy třeba obejít tuto inkompatibilitu pomocí

„SET“ přídavku k „LOAD DATA“ příkazu:

```
      SET  datum = STR_TO_DATE(@datum, '%d.%m.%Y') ;
```

a celý příkaz poté vypadal pro tabulku „pozvanky“ takto:

```
echo „LOAD DATA LOCAL INFILE '/tmp/komplet-utf8/pozva00135.unl'
 INTO TABLE  pozvanky
  CHARACTER SET utf8
    FIELDS TERMINATED BY '|'
   ( @datum,  cdar,  hodina,  centrum,  typo,  nahradnik )
    SET  datum = STR_TO_DATE(@datum, '%d.%m.%Y') ;“  \
| mysql -uroot -ppwd  isto_ov
```

a nebo u tabulek s větším počtem polí i takto :

```
select now() as 'About to load vanda' ;
explain select  now()  from vanda ;
LOAD DATA LOCAL INFILE '/tmp/komplet-utf/vanda00343.unl'
 INTO TABLE  vanda
  CHARACTER SET utf8
   FIELDS TERMINATED BY '|'
    (  codb,  vysp,  tod,  nazs,  roz,  pr,  del,  kr_sk,  rh_f,
@dexp,  hexp,  rodne,  komu,  tu,  mnoz,  mnoz_ml,  d_list,  cod,
dv_list,  menv, @datv,  casv, @datvr,  dovvr,  menp, @datp,  casp,
cd,   svysp,   cisfa,  @datfa,  @dspfa,   menzr,  @datzr,   dovzr,
centrum )
  SET  dexp  = STR_TO_DATE(@dexp,  '%d.%m.%Y'),
       datv  = STR_TO_DATE(@datv,  '%d.%m.%Y'),
       datvr = STR_TO_DATE(@datvr, '%d.%m.%Y'),
       datp  = STR_TO_DATE(@datp,  '%d.%m.%Y'),
       datfa = STR_TO_DATE(@datfa, '%d.%m.%Y'),
       dspfa = STR_TO_DATE(@dspfa, '%d.%m.%Y'),
       datzr = STR_TO_DATE(@datzr, '%d.%m.%Y') ;
```

Viz příloha 1: program, který vygeneroval všechny tyto SQL příkazy.

7.1 Práce s vlastními daty v databázi ELAS

Protože zde nebyla garance, že všechny sloupce byly na úrovni databáze propojeny formou cizího klíče (foreign key) s ostatními sloupci, které nesou příslušné hodnoty, bylo nutno v rámci detektivní práce takové sloupce a tabulky dohledávat podle známých hodnot. Například se zjistilo, že číslo **12111303** je číslem spojeným s odběrem krve, ale nevědělo se kde a jak. Tak bylo třeba najít, kde všude se vyskytuje v databázi, a prozkoumat, zda-li mezi těmito sloupci existuje konkrétní aplikační pravidlo („business rule“):

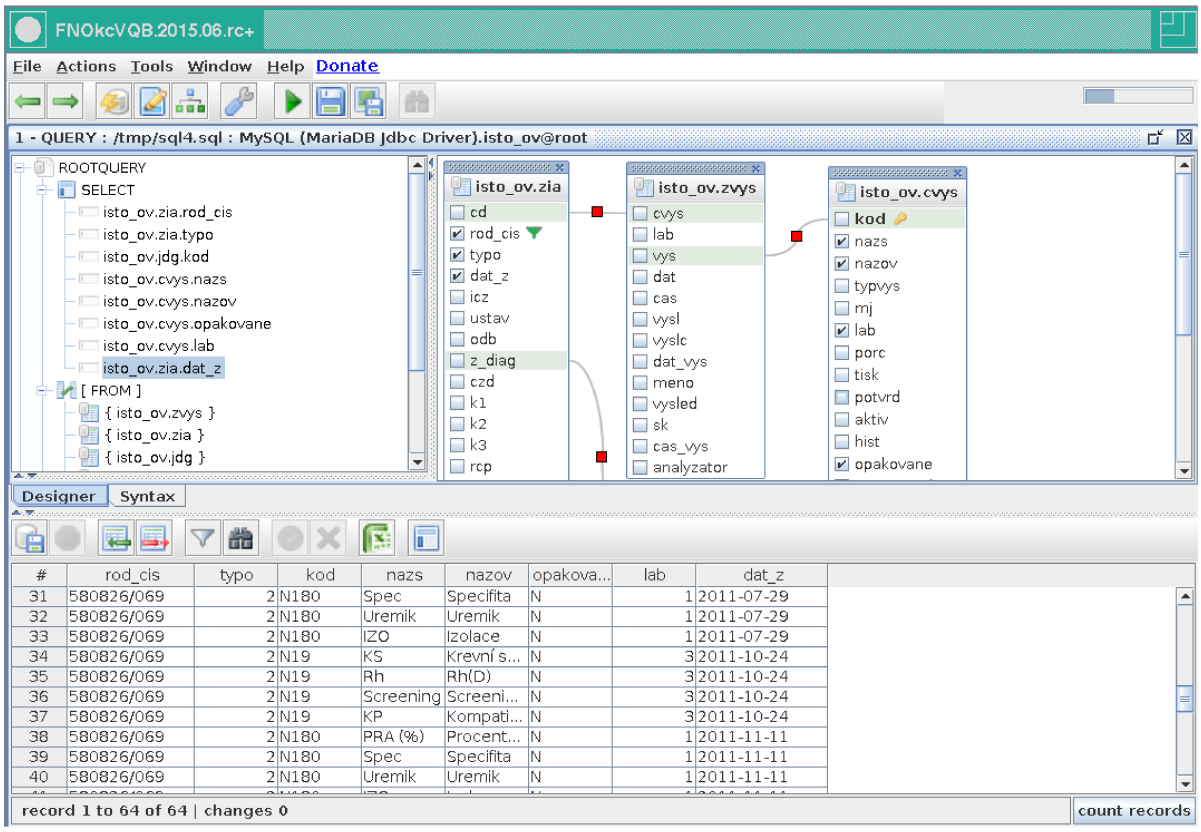
Dohledávání „který sloupec je na aplikační úrovni spojen s kterým“ :

```
for i in \
  $( echo $' use isto_ov ; show tables ; ' \
    | mysql –uroot –ppwd ) ; # seznam všech tabulek
do echo =========== $i ;
   echo $' use isto_ov ;
   show create table '$i' ; ' \
  | mysql –u root –ppwd ; \
done \
| perl –pe 's/\\n/\n/g' \
| perl –ne '/CREATE TABLE `(\S+)`/ and  $t=$1;
  /`(\S+)` (char\((\d+)|decimal\((\d+))/
  and ( $3 > 8 or $4 > 8 )
  and  print "$t\t$1\t$3\t$4\t$2$/"' \
| while read t f j ; \
 do echo "explain select * from $t where $f = 12111303 ; " ;
    echo "        select * from $t where $f = 12111303 ; " ; \
done | mysql –u root –ppwd  –D isto_ov   | grep –iv possible_keys
```

8 Popis nástroje FNOkcVQB

V této sekci se věnuji popisu nástroje FNOkcVQB, který slouží pro vizuální interaktivní tvorbu dotazů na anonymizovanou ELAS databázi a dle požadavků Krevního centra následného exportu do spreadsheetu (Excel, CSV formát).

Tato sekce nemíní plnohodnotně nahradit manuál, který může být případně pro KC FNO vytvořen. Jen popisuje nezbytnou orientaci v programu.



Obr. 3 Nejčastější/nejvíce používaná konfigurace oken pro hlavní účel FNOkcVQB

8.1 Obecný popis programu

FNOkcVQB pracuje nad databází, a to i nad několika schématy najednou.

FNOkcVQB umožňuje současně editovat a spouštět několik nezávislých dotazů, každý ve svém okně. Dotazy mohou být ukládány na disk a později buď nahrány zpět k další editaci nebo spuštěny přímo v MySQL jako externí SQL soubor.

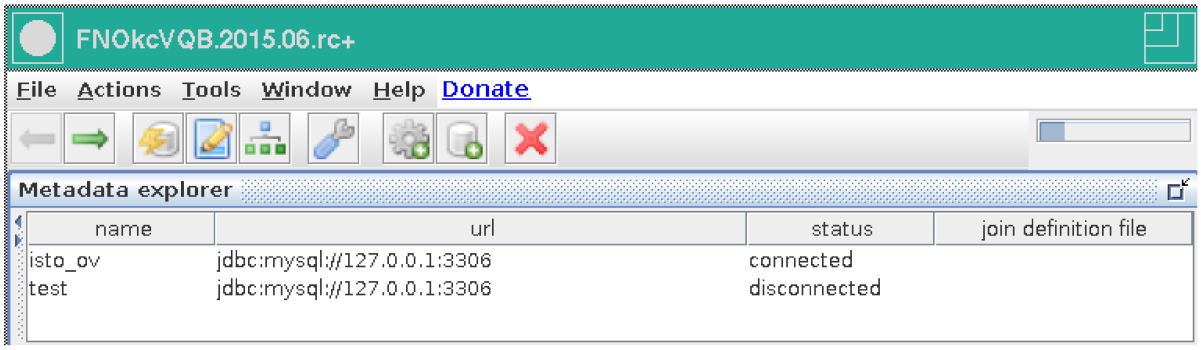
FNOkcVQB si i po ukončení pro pozdější použití drží na disku základní historii dotazů zaslaných databázi a připojovací data a konfigurační data.

FNOkcVQB má tři hlavní nástroje:

1. Metadata explorer – jedno okno
2. Editor příkazů (command editor) – jedno okno
3. Tvorba dotazů (Query builder) – jedno okno pro každý editovaný dotaz

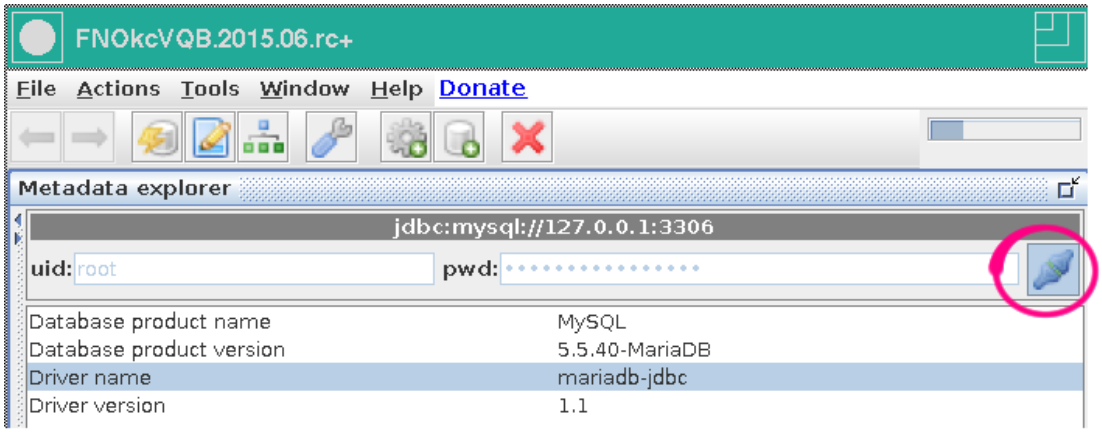
8.2 Metadata explorer

Po spuštění nás přivítá okno metadata exploreru, oznamující, ke které databázi máme přednastavené připojení:



Obr. 4 Metadata explorer

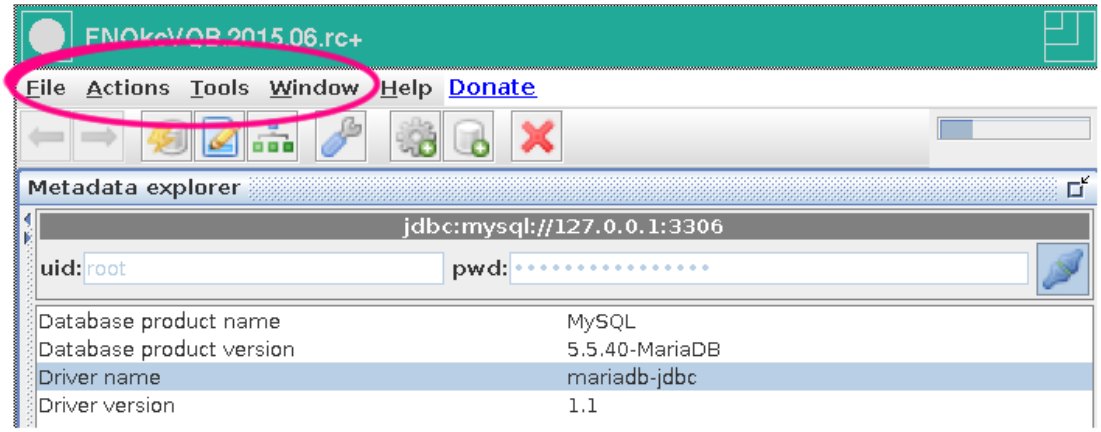
nebo ke které databázi jsme připojeni:



Obr. 5 Připojení k databázi

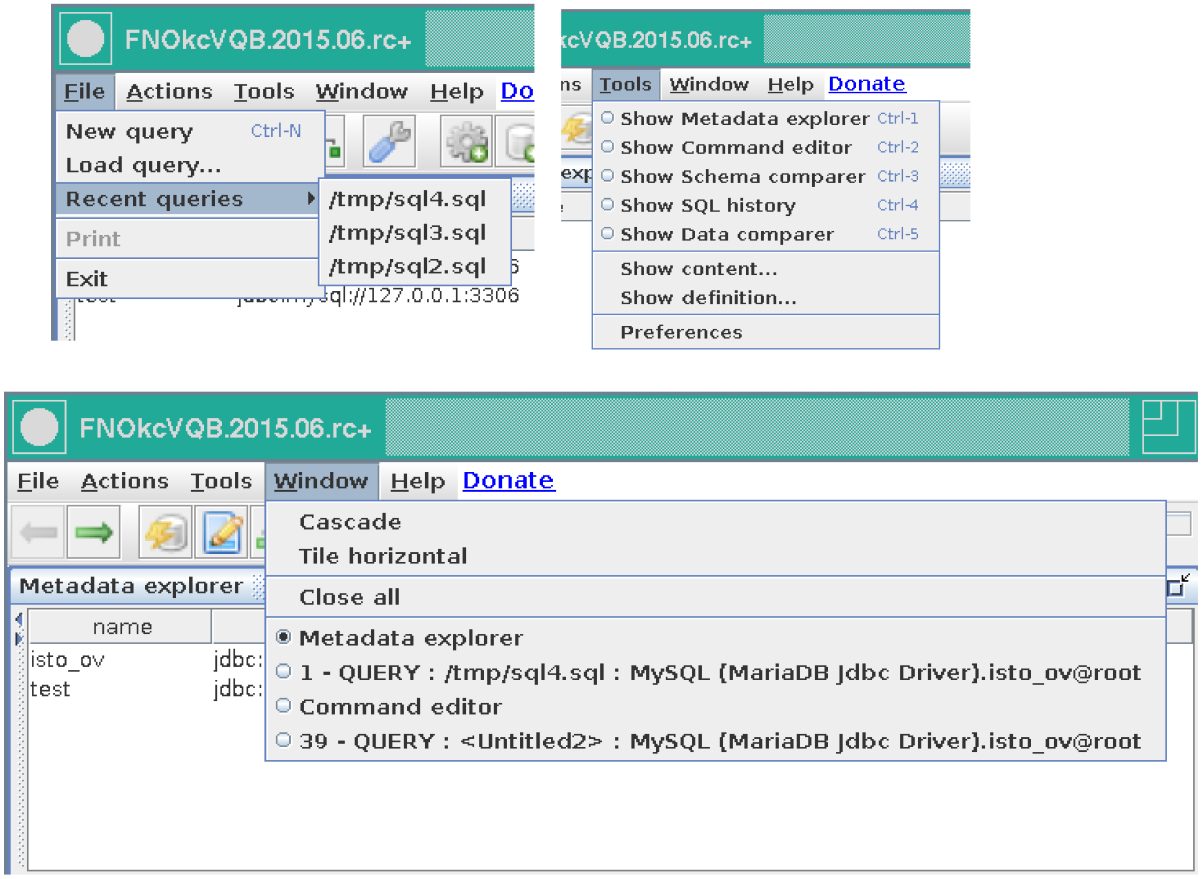
Stručný popis hlavního menu a ikon, které nás provází celým programem.

Menu:



Obr. 6 Hlavní menu

V průběhu programu pracujeme se třemi fixními sadami nabídky (File, Tools, Windows) a jednou proměnlivou (Actions) v závislosti na momentálním výběru nástroje.

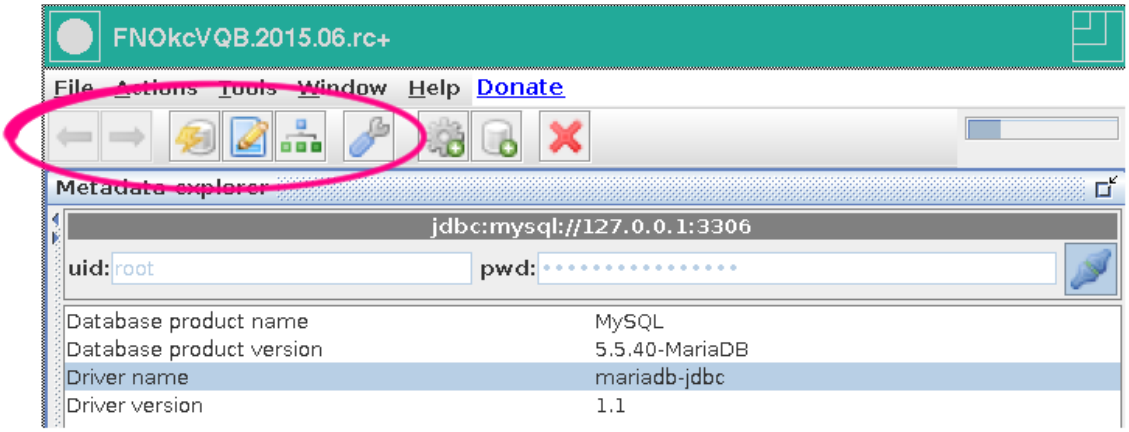


Obr. 7-9 Nabídka

Obsah nabídky File je samo popisující, u nabídky Tools je snad akorát třeba podotknout, že funkčností nástrojů „Schema comparer“, „Data comparer“ a „SQL History“ se zde nezabýváme, a nabídka Window je zde zcela standardním GUI elementem, taktéž jej není třeba popisovat.

8.3 Výběr nástroje

Prvních 6 ikonek zleva jsou fixní pro celou aplikaci.



Obr. 10 Ikony

První dvě ikony umožňují pohyb v rámci již otevřených oken nástrojů, třetí ikona přepíná přímo do metadata exploreru, čtvrtá přímo do command editoru, pátou editujeme preference a šestá nám otevře nové okno pro editaci nového dotazu.

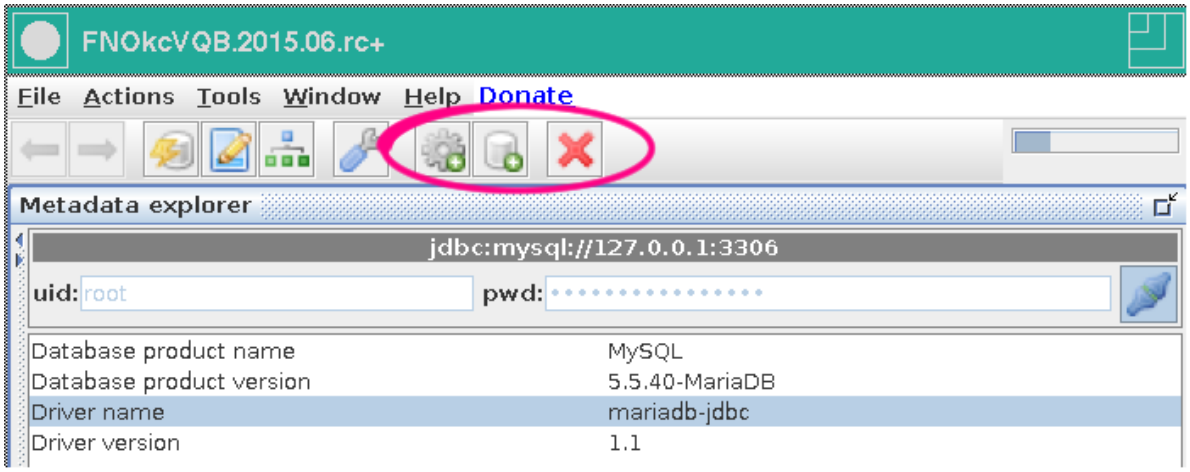
8.4 Připojení k databázi

Pro připojení k databázi a pro práci s daty v ní uloženými je obecně třeba znát několik nezbytných údajů. Těmito jsou:

1. jméno nebo IP adresa počítače, na kterém běží daný systém řízení báze dat
2. TCP Port, skrze který je na tomto počítači přístupný (pro MySQL je to standardně 3306)
3. uživatelské jméno vytvořené v databázi (není nezbytně shodné s uživatelským jménem, pod kterým běží systém řízení báze dat či jiným uživatelským jménem rozpoznaným v OS počítače).
4. heslo pro autentizaci DB uživatele z bodu 3
5. jméno databázového schématu, pod kterým jsou požadovaná data přístupná (v našem případě přednastaveno na isto_ov)
6. typ systému řízení báze dat (v našem případě MySQL/MariaDB)

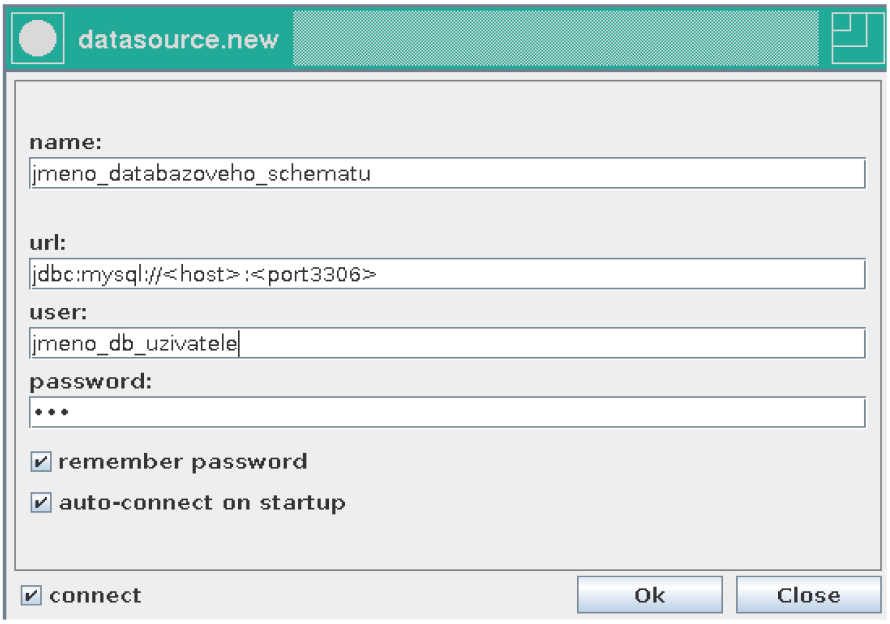
Výše uvedené body 1. až 4. jsou specifické pro tu kterou instalaci databáze, je tedy třeba, aby byly poskytnuty tím, kdo databázi instaloval na ten který počítač/server.

Takovéto spojení lze potom nastavit skrz nástroj metadata explorer, kliknutím na druhou ikonku zprava: „New datasource...“



Obr. 11 Připojení k databázi

Což nám umožní vložit údaje pro nové připojení k databázi:



Obr. 12 Vložení údajů pro nové připojení

Zde url pro testovací účely může vypadat třeba takto: jdbc:mysql://127.0.0.1:3306 .

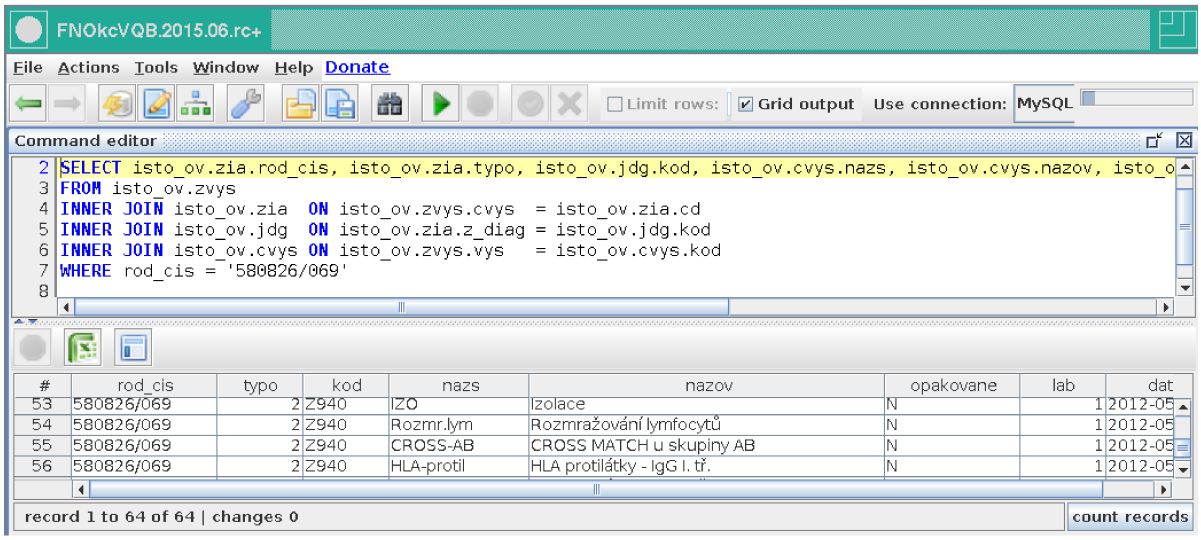
V případě, že jsme něco nezadali správně nebo příslušná databáze není správně nainstalována, dostaneme po kliknutí OK hlášku:

Obr. 13 Chybová hláška

V opačném případě se nám v přehledu spojení objeví „connected“ na příslusném řádku. Viz obrázek „Přehled připojení“ výše.

8.5 Command editor

Command editor nám umožňuje zadávat SQL příkazy přímo bez použití grafické tvorby. Dává nám tím o něco vyšší pružnost ve vyjadřování dotazů, ale je použitelný primárně osobami s dobrou znalostí SQL jazyka.



Obr. 14 Command editor

Vedle šesti fixních ikon se nalézají, v pořadí :

- Otevřít soubor s SQL dotazem
- Uložit soubor s SQL dotazem
- Vyhledat řetězec v textu dotazu a zaměnit části textu dotazu
- Spustit dotazovací
- Zastavit spuštěný dotaz

a dvě funkce, které na této úrovní datawarehosingu/data miningu obecně nepoužíváme :

- Commit a Rollback

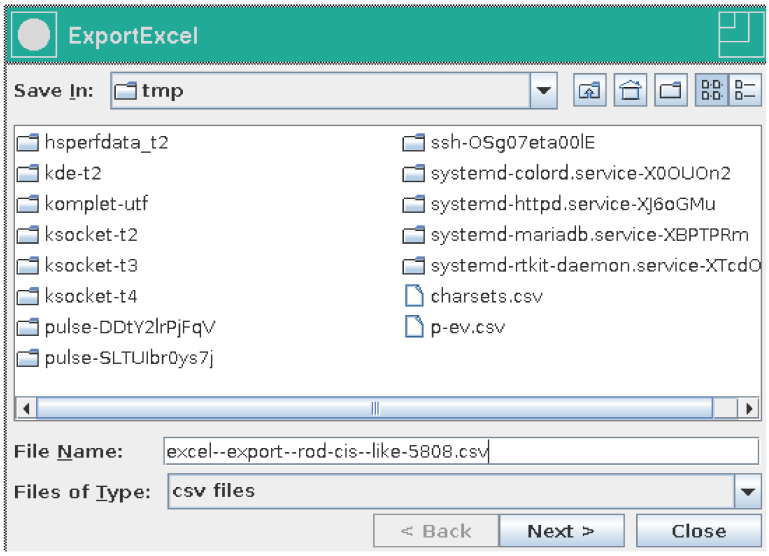
Vpravo nahoře je též možno zakřížkovat „Grid output“, což způsobí, že ve spodní části okna se neobjeví standardní textový výstup databáze, ale tabulkový. Tento výstup je pak možno dokonce editovat a zapsat zpět do databáze, takže je třeba si dávat pozor, abychom něco omylem nepřepsali.

Ikonky mezi textem dotazu a gridem pak umožnují, v pořadí:

1. Zastavit spuštěný dotaz
2. Exportovat výsledek dotazu do spreadsheetu, ve formátu CSV
3. Vytvořit pivot tabulku

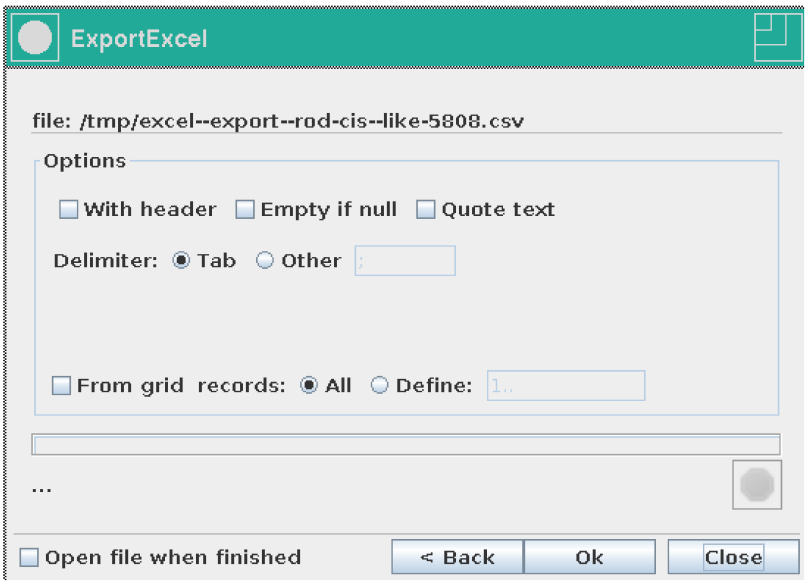
Funkce „Pivot tabulka“ je zde pro případ, že nemáme právě po ruce spredsheet program, kterým bychom si mohli exportovaná data prohlédnout a prozkoumat, ale máme poruce prohlížeč Firefox nebo Chrome. Kliknutím se nám v takovém prohlížeči spustí, na poměry internetového prohlížeče, překvapivě velmi bohaté uživatelské rozhraní pro různé grafické i tabulkové zobrazení takto získaných dat.

Funkce Exportu do .CSV nám umožní, kromě samozřejmého místa a názvu souboru uložení dat.



Obr. 15 Export do .csv

Po kliknutí Next také můžeme dopřesnit jejich CSV formát:

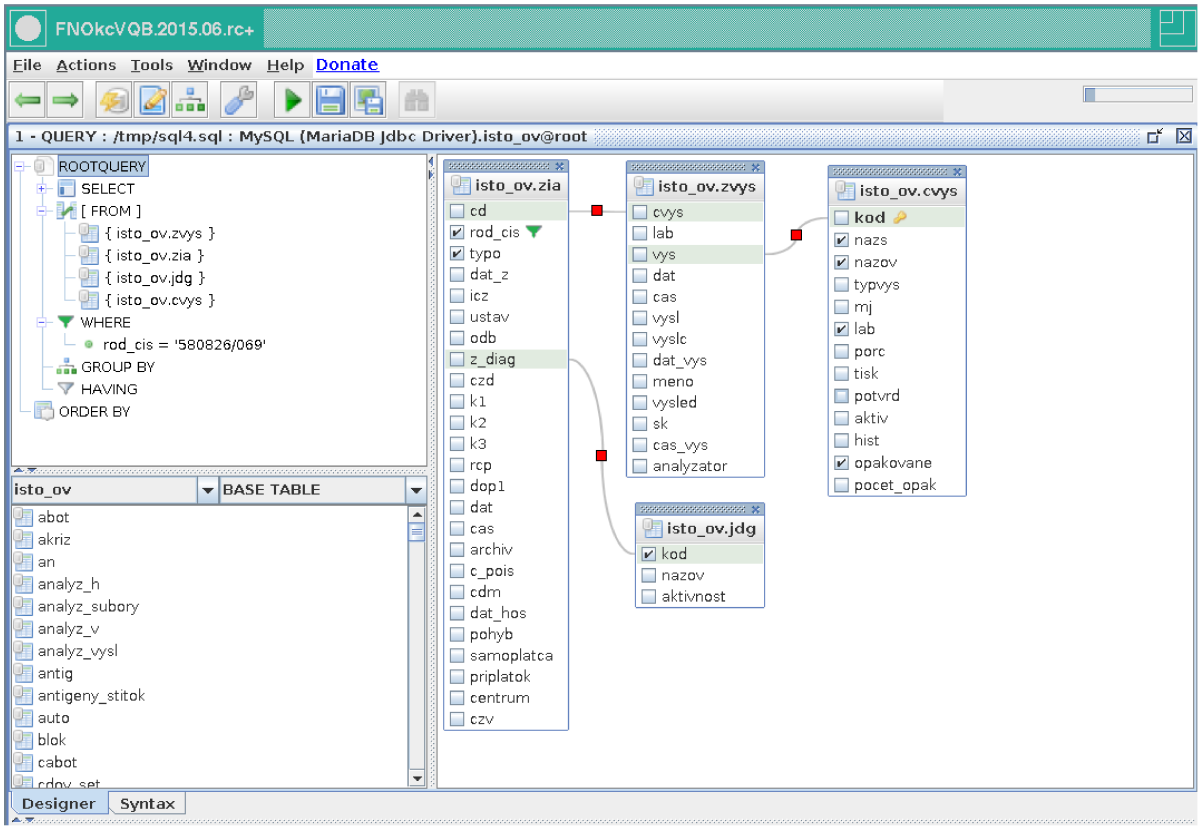


Obr. 16 Dopřesnění CSV formátu

8.6 Vizualní tvorba dotazu

Tento nástroj je ústředním nástrojem pro použití v KC FNO. Naštěstí je poměrně intuitivní.
Tímto nástrojem si personál KC FNO bude moci vytvářet své hlavní dotazy na medicínské data původně uložené v databázi ELAS a následně dle požadavků KC FNO exportovat do .CSV formátu pro další potřeby.

Typický obsah okna při vytváření dotazu takového nástroje vypadá asi takto:



Obr. 17 Vytváření obsahu

Zde máme tři hlavní panely pro tvorbu. Z levého spodního vybíráme relace (tabulky), které se nějak účastní na tvořeném dotazu. Úkonem „táhni-a-pusť“ přetáhneme takto vybrané tabulky jednu za druhou do pravé části a graficky umístíme pro pohodlné zobrazení. Pak jiným úkonem „táhni-a-pusť“ spojíme v pravé části jednotlivá pole relací, u kterých známe vztah rovnosti. Ve zde vyobrazeném dotazu je isto_ov.zvys.cvys = isto_ov.zia.cd , isto_ov.zia.z_diag = isto_ov.jdg.kod a isto_ov.zvys.vys = isto_ov.cvys.kod.

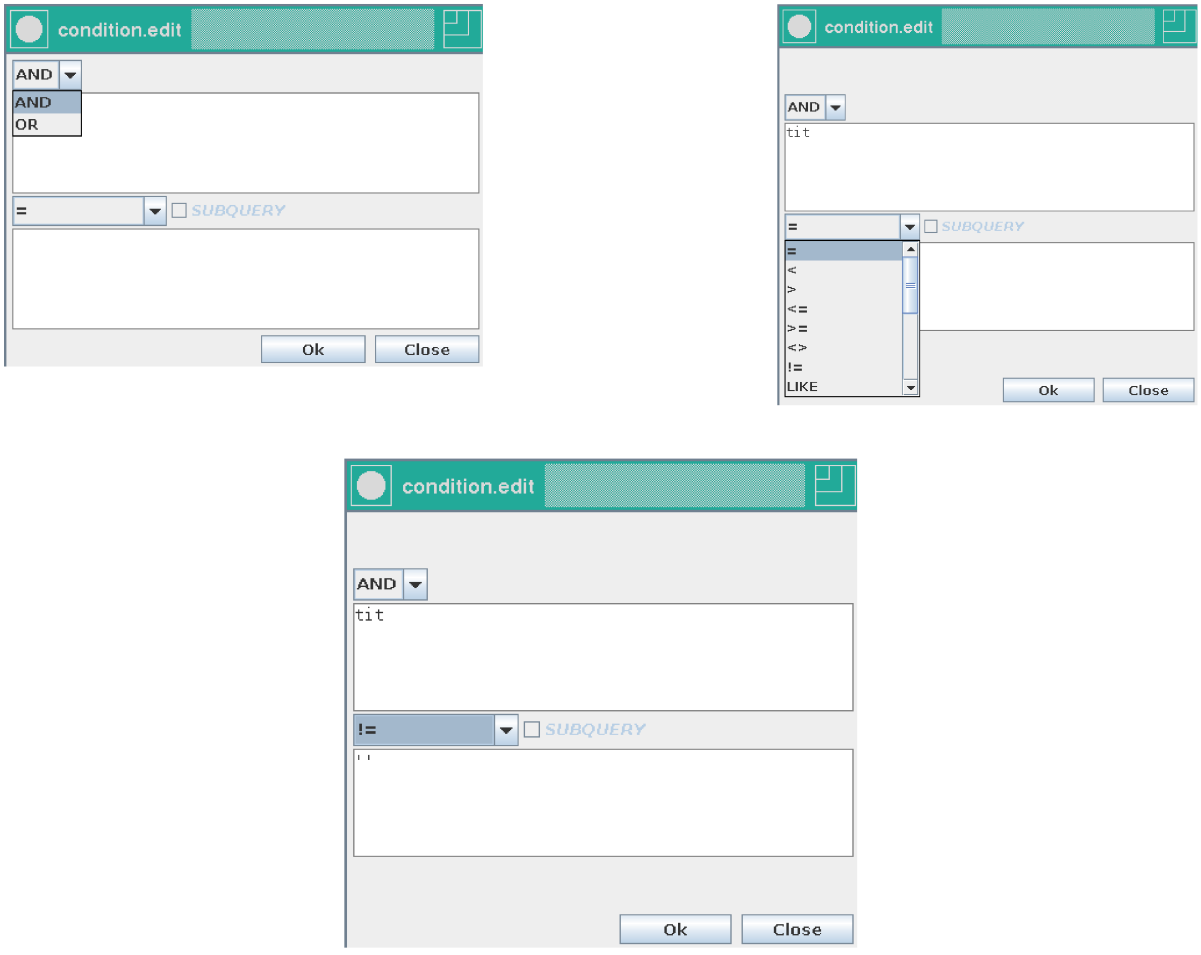
V pravé části si dále vybereme zaškrtnutím příslušných polí, které údaje chceme výsledným dotazem dostat/zobrazit.

V levém horním panelu je pak vidět struktura takto tvořeného dotazu. Skrze tento panel můžeme vytvářet i poddotazy, přidávat podmínky výběru řádků i group a označit způsob řazení výsledků.

Vedle šesti fixních ikonek aplikace vidíme další 4 pro tento nástroj:

- spusť tvořený dotazu
- uschování tvořeného dotazu na disk
- uschování screenshotu tvořeného dotazu na disk
- Find & Replace v textu dotazu, který můžeme vidět, když se vlevo dole přepínáme mezi „Designer“ (graficky zobrazený dotaz) a „Syntax“ (textově zobrazený dotaz).

Přidávání podmínek se děje buď textově (pro ty zkušenější) v panelu Syntax nebo graficky pravým kliknutím na „WHERE“ a vybráním „Add condition ...“, pak v případě druhé a dalších podmínek je třeba vybrat, zda-li má být logicky konjuktivní (AND) nebo aditivní (OR), vybrat pole nebo výraz v dotazu a hodnotu, se kterou se každý takto vybíraný řádek má porovnávat.



Obr. 18 – 20 Přidávání podmínek

Kliknutím na „Spusť“ se pak otevře na spodní části panel výsledků, ve které vidíme kromě dat databází na dotaz vybraných i 10 ikonek pro spuštění některých dalších funkcí :

- ikonky 1,2,3,4,7, a 8 jsou pro provádění změn v záznamech databáze a proto se zde jimi nezabýváme.
- ikonka pátá je pro další filtrování řádků, tedy trochu jiný způsob zadávání podmínek do WHERE klauzule
- ikonka šestá se používá pro vyhledávání hodnot ve výsledcích dotazů a případné záměně za hodnoty jiné.

Funkce posledních dvou ikonek je již popsána v sekci „Command editor“. Zde se export do spreadsheetu použije jako hlavní výstupní krok při tvorbě dotazu. Dotaz samotný se ale také může uložit pro pozdější použití (třetí ikonka zprava ve vrchním panelu).

Závěr

Proces tvorby této diplomové práce byl zajisté zkušeností otevírající nové obzory. Z hlediska problematiky nástrojů pro data mining jsem dospěla k názoru, že věci vůbec nejsou tak jednoduché, jak se zprvu jeví a že množství detailů a jejich různorodosti, se kterými se úspěšný data miningový nástroj musí potýkat, zakládá takovým nástrojům váženou pozici na trhu. Když si dnešní firmy účtují milióny za licence i pro ty omezenější produktové řady data mining nástrojů, tak se zdá jsou zcela v právu.

I pro náklady s data miningem spojené si tradičně vyspělá data miningová oddělení mohly do svých podnikatelských struktur plnohodnotně zařadit jen firmy s velkým objemem obchodu, typicky s tisíci či aspoň stovkami zaměstnanců. Avšak je trendem posledních let stále častěji uplatňovat data mining pro své procesy i v menších společnostech čítajících jen desítky zaměstnanců, protože se na trhu stále objevují nové data miningové nástroje nejen s vlastnostmi specifickými pro ten který obor podnikání, ale i s přitažlivými licenčními podmínkami.

Výsledkem této práce je dodání variabilního nástroje adaptovaného pro řešení konkrétních problémů specifických pracovních a vědeckých procesů Krevního centra FNO. Ale v těchto procesech se nachází i komplexnější a hlubší problémy z oblasti data miningu, které zatím tato verze implementace daného nástroje neřeší zcela úspěšně. Podstatných vylepšení by bylo možno dosáhnout zejména typickými pokročilými data miningovými úkony, jako je denormalizace dané báze dat a optimalizace s podporou pro pokročilé dotazy. Z hlediska uplatnění tohoto nástroje se pak naskýtá i možnost jeho průběžného užití vůči novému informačnímu systému Krevního centra, tedy vůči nástupci systému ELAS. Toto by už ale byl další, i když navazující, projekt.

Dalším krokem, kterým by se daná práce mohla vyvíjet, je připojení obecně dostupných informací, jako je například goegrafická poloha, historické záznamy kvality ovzduší nebo vodních zdrojů, objemy dovozů potravin z jiných států, apod. a následné hledání vzájemných souvislostí.

Literatura

[1] SKLENÁK, Vilém. *Data, informace, znalosti a Internet*. Vyd. 1. V Praze: C.H. Beck, 2001, s. 2. C.H. Beck pro praxi. ISBN 80-7179-409-0.

[2] ŽÁK, Karel. *Historie relačních databází.* In: Root.cz [online]. 2001 [cit. 2015-01-05]. Dostupné z: http://www.root.cz/clanky/historie-relacnich-databazi/

[3] POKORNÝ, Jaroslav. *Databázové systémy a jejich použití v informačních systémech*. 1. vyd. Praha: Academia, 1992, ISBN 80-200-0177-8.

[4] Databázové modely. *Databáze* [online]. 2010 [cit. 2015-07-30]. Dostupné z: http://databaze.chytrak.cz/modely.htm

[5] SKŘIVAN, Jaromír. *Databáze nejsou jen MySQL*. In: Interval.cz [online]. 2002 [cit. 2015-01-05]. Dostupné z: http://interval.cz/clanky/databaze-nejsou-jen-mysql/

[6] Úvod do databází. *Databáze* [online]. 2010. [cit. 2015-07-30]. Dostupné z: http://databaze.chytrak.cz/index.htm

[7] HORÁK, Jiří a Bronislava HORÁKOVÁ. *Datové sklady a využití struktury hvězda pro prostorová data* [online]. : 26. [cit. 2015-07-30]. Dostupné také z: http://gis.vsb.cz/GIS_Ostrava/GIS_Ova_2007/sbornik/Referaty/Sekce3/hvezdaF4.pdf

[8] DANIEL, Roman. *Datový sklad* [online]. Ostrava, 2010 [cit. 2015-07-30]. Dostupné z: http://homel.vsb.cz/~dan11/is_skripta/IS%202010%20-%20Danel%20-%20Datovy%20sklad.pdf

[9] STĚHULE, Pavel. PostgreSQL. *Postgres* [online]. [cit. 2015-07-30]. Dostupné z: http://postgres.cz/wiki/PostgreSQL

[10] ,. Oracle (databázový systém). *Wikipedia* [online]. [cit. 2015-07-30]. Dostupné z: https://sk.wikipedia.org/wiki/Oracle_%28datab%C3%A1zov%C3%BD_syst%C3%A9m%29

[11] ZAJÍC, Petr. MySQL (1) - pestrý svět databází. *Linuxsoft.cz* [online]. 2005 [cit. 2015-07-30]. Dostupné z: http://www.linuxsoft.cz/article.php?id_article=731

[12] JANDA, Jaroslav. *Jemný úvod do SQL*. 1997. [cit. 2015-07-30]. Dostupné také z: http://www.wlist.cz/subsql/SQL.pdf

[13] PALOVSKÁ, Helena. SQL. *Krokodýlovy databáze* [online]. 2014 [cit. 2015-07-30]. Dostupné z: http://krokodata.vse.cz/SQL/SQL

[14] Právní aspekty: Anonymizace. In: HLADKÁ, Eva a Jan FOUSEK. *Základy IT gramotnosti* [online]. [cit. 2015-01-21]. Dostupné z: http://is.muni.cz/do/1492/el/sitmu/law/html/ch02s10.html

[15] § 4 písm. a) zákona č. 101/2000 Sb., o ochraně osobních údajů

[16] KLÍMA, Vlastimil. *Tunely v hašovacích funkcích: kolize MD5 do minuty*. 2006. [cit. 2015-07-

30]. Dostupné také z: http://cryptography.hyperlink.cz/2006/tunely.pdf
[17] PROCHÁZKA, Michal. Data mining: jiný pohled na problém. *VTM* [online]. [cit. 2015-07-31]. Dostupné z: http://vtm.e15.cz/aktuality/data-mining-jiny-pohled-na-problem
[18] BÁRTÍK, František. KNIME: nástroj pro analýzu rozsáhlých dat. *Linuxexpres.cz* [online]. 2014 [cit. 2015-07-31]. Dostupné z: http://www.linuxexpres.cz/software/knime-nastroj-pro-analyzu-rozsahlych-dat
[19] DELL. *Statsoft.com* [online]. 2015 [cit. 2015-07-31]. Dostupné z: http://www.statsoft.com/
[20] *Grafické rozhraní: Úvod do tvorby GUI* [online]. In: . [cit. 2015-07-31]. Dostupné z: http://www.temp.buchtic.net/skola/ipogr/prednasky/IPOGR_2010_P02_Java-GrafickeRozhrani_4s.pdf
[21] DOSTÁL, Martin. *Základy tvorby uživatelského prostředí* [online]. In: . Olomouc, 2007 [cit. 2015-07-31]. Dostupné z: http://phoenix.inf.upol.cz/esf/ucebni/gui-dostal.pdf

Přílohy

Příloha A, 1 strana :

Program pro vygenerování SQL LOAD příkazů pro všechny tabulky

Příloha B, 2 strany :

Výsledný program, který procesoval vyhledané sloupce k anonymizaci

Příloha A:

Program pro vygenerování SQL LOAD příkazů pro všechny tabulky

```
perl -ne '/TABLE (.*) row size =/ and $next_unload=1 and $t=$1;
/unload file name = (\S+)/ and $next_unload==1 and print "$1\t$t$/"
and $next_unload=0 ' /cesta/k/elas/dump/isto_ov.sql \
| grep -v '"' \
| while read fn t ; \
  do    echo $'
    use isto_ov ;
    select "filename = '$fn'" as "" ;
    show create table '$t' ; ' ;
  done \
| mysql -u root -ppwd \
| perl -pe 's/\\n/\n/g' \
| perl -ne 'END{ outme() };
 BEGIN{ sub outme{ return 1 if $fn eq "";
   $loadstr = "
 select now() as \x27About to load $t\x27 ;
 explain select  now()  from $t ;
 LOAD DATA LOCAL INFILE \x27/tmp/komplet-utf/$fn\x27
 INTO TABLE  $t
 CHARACTER SET utf8
 FIELDS TERMINATED BY \x27|\x27
  ( ".join(", ", @fields)." )";
$sets=""; if (@sets) { $sets=" SET ".join(", ", @sets) };
$loadstr  .=  $sets."  ;  $/";  @sets=@fields=@dates=();  print
"$loadstr" ;  1;}; }; # end sub outme()
 /^  `(\S+)` (date )?/ and ($f,$d)=($1,$2)
 and "print $f\t$d\t$fn\n"
 and $p=($2 eq "date " ? "@" : " ")
 and push(@fields, $p.($f=$1) )
 and do {  if ($p eq "@") {
   push(@dates, $f);
   push(@sets," $f = STR_TO_DATE(\@".$f.", \x27%d.%m.%Y\x27)");}};
 /CREATE TABLE `(\S+)`/ and $t=$1;
 /^filename = (\S+)/
  and $newfn=$1
  and outme()
  and $fn=$newfn; ' > filename-tablename.txt.load.sql
```

Příloha B :

Výsledný program, který procesoval vyhledané sloupce k anonymizaci vypadal takto :

```
#!/bin/bash

EXPORTDIR=/cesta/k/db/dumpu/isto_ov.exp  #  vstup
ANONYMDIR=/tmp/vystup/1                  # vystup

cd $EXPORTDIR

KLIC=/tmp/isto_ov.perf_klic.dat

anonymize_export () {

perl -e '

$exportdir = "'$EXPORTDIR'";
$anonymdir = "'$ANONYMDIR'";

$keyfile = "'$KLIC'";

# tabulky a jejich sloupce, které obsahují rodná čísla
@tables_cols_orig = qw(
akriz00230.unl 1
auto_00129.unl 0
 # … dalších několik desítek tabulek
zia__00364.unl 1   12
);

sub doit {

  if ( $perfect_key_done ) {
        open(KLIC, "<$keyfile") or die "nelze otevrit klic : $keyfile $/";
        while(<KLIC>) {
                chomp;
                  @F = split ("\\|", $_); # double backslash: we are perl
inside shell
                $perf_key{$F[0]} = $F[1];
#               print "$F[0]\t".$perf_key{$F[0]}.$/;
        };
  } else {
          open(K, ">$keyfile") or die "cannot open : gzip -9 >$keyfile $/"
  };

  @tables_cols = @tables_cols_orig;
  while ( @tables_cols ) {
        $c2 = undef;
        ($t, $c) = ( shift @tables_cols, shift @tables_cols );
        $c2 = shift @tables_cols
                if $tables_cols[0] =~ /^\d+$/;
        print "$t\t$c\t$c2\n";
#       warn "opening : zcat $exportdir/$t.gz $/";
          open(T, "zcat $exportdir/$t.gz |") or die "cannot open : zcat
$exportdir/$t.gz ";
          open(A, "| gzip -9 >$anonymdir/$t.gz |") or die "cannot open :
gzip -9 >$anonymdir/$t.gz "
                if $perfect_key_done;
        while (<T>) {
                @F = split ("\\|", $_);  # double backslash again
```

```
                if ( ! $perfect_key_done ) {
                        chomp $F[$c];
                        $F[$c]=~ m/((()))/;
                        $F[$c]=~m:^((\d{6})/(.*))$:;
                        $rc=$1;
                        $d=( $2 ? $2 : $dc++ );
                        $p=( $3 ? $3 : $pc++ );
                        if ( ! $perf_klic{$rc} ) {
                                $perf_klic{$rc} = $new_key = sprintf("%06d/
%03d", $d, $counter{substr($2,0,4)}++ );
                          print K "$F[$c]|$new_key|$/";
                        }
                        next unless $c2;
                        $c = $c2;
                        chomp $F[$c];
                        $F[$c]=~ m/((()))/;
                        $F[$c]=~m:^((\d{6})/(.*))$:;
                        $rc=$1;
                        $d=( $2 ? $2 : $dc++ );
                        $p=( $3 ? $3 : $pc++ );
                        if ( ! $perf_klic{$rc} ) {
                                $perf_klic{$rc} = $new_key = sprintf("%06d/
%03d", $d, $counter{substr($2,0,4)}++ );
                          print K "$F[$c]|$new_key|$/";
                        }
                        next unless $c2;
                        $c = $c2;
                        chomp $F[$c];
                        $F[$c]=~ m/((()))/;
                        $F[$c]=~m:^((\d{6})/(.*))$:;
                        $rc=$1;
                        $d=( $2 ? $2 : $dc++ );
                        $p=( $3 ? $3 : $pc++ );
                        if ( ! $perf_klic{$rc} ) {
                                $perf_klic{$rc} = $new_key = sprintf("%06d/
%03d", $d, $counter{substr($2,0,4)}++ );
                          print K "$F[$c]|$new_key|$/";
                        }
                } else {
                $F[$c]  = $perf_key{$F[$c]};
                $F[$c2] = $perf_key{$F[$c2]} if $c2;

                 if ( $t eq "p_ev_00385.unl" ) { # jmeno, prijmeni, adresa,
telefon. Ale PSC(F[7]) zustava.
                        $F[1] = $F[2] = $F[5] = $F[6] = $F[8] = ""; };
                # pro další tabulky obdobně
                print A  join( "|", @F );
                };

        };
} # while
$perfect_key_done = 1; close K;
} # doit

doit(); # make peftect key
doit(); # anonymize
';
}
```